МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе 7**

по дисциплине: «Программирование на языке Java»

Выполнили: студенты гр. 10702122 Тимощенко Н.В   
 Кудрявец Д.Е

Проверила: пр.- ст. Савчик А.

Минск 2024

**Лабораторная работа №7.**

**Цель работы:** изучить работу наследования, композиции, агрегации.

**Задание 1:**

Изучите и законспектируйте теорию:

1. Агрегация

2. Наследование

Самостоятельно:

Реализовать код приведенный в разделе про агрегацию

Написать javadoc комментарий к классу и методам.

**Код программы:**

*/\*\*  
 \* Проверка классов  
 \*  
 \* @author Tsimoshchenko M.V, Kudryavets D.E  
\* @version 1.0  
 \*/*public class AggregationExample {  
 public static void main(String[] args) {  
 *// Создание статей* Note\_ noteVasilev1 = new Note\_("Васильев П.И.", "Создание классов в Java", 3.6);  
 Note\_ noteVasilev2 = new Note\_("Васильев П.И.", "Среда NetBeans", 2.8);  
  
 *// Создание массива статей* Note\_[] notes = new Note\_[10];  
 notes[0] = noteVasilev1;  
 notes[1] = noteVasilev2;  
  
 *// Создание журнала* String PUBLISHING\_HOUSE = "Эрудит";  
 String PUBLISHING\_LOCATION = "Минск";  
  
 Magaz magazine = new Magaz("Программирование", 1, 2016,  
 notes, PUBLISHING\_HOUSE,  
 PUBLISHING\_LOCATION);  
 System.*out*.println(magazine.toString());  
 }  
}  
  
*/\*\*  
 \* Класс для представления журнала  
 \*/*class Magazine {  
 private String magazineTitle; *// название журнала* private int numberMagazine; *// номер выпуска* private int publishingYear; *// год издания* private Note\_[] notes; *// массив статей* private String publishingHouse; *// издательство* private String publishingLocation; *// город* private int numberPage; *// количество страниц  
  
 /\*\*  
 \* Конструктор для создания экземпляра журнала  
 \* @param magazineTitle - название  
 \* @param numberMagazine - номер выпуска  
 \* @param publishingYear - год издания  
 \* @param notes - массив статей  
 \* @param publishingHouse - издательство  
 \* @param publishingLocation - город  
 \*/* public Magazine(String magazineTitle, int numberMagazine,  
 int publishingYear, Note\_[] notes,  
 String publishingHouse, String publishingLocation) {  
 this.magazineTitle = magazineTitle;  
 this.numberMagazine = numberMagazine;  
 this.publishingYear = publishingYear;  
 this.notes = notes;  
 this.publishingHouse = publishingHouse;  
 this.publishingLocation = publishingLocation;  
 }  
  
 *// Геттеры для всех полей класса* public String getMagazineTitle() {  
 return magazineTitle;  
 }  
  
 public int getNumberMagazine() {  
 return numberMagazine;  
 }  
  
 public int getPublishingYear() {  
 return publishingYear;  
 }  
  
 public Note\_[] getNotes() {  
 return notes;  
 }  
  
 public String getPublishingHouse() {  
 return publishingHouse;  
 }  
  
 public String getPublishingLocation() {  
 return publishingLocation;  
 }  
  
 public int getNumberPage() {  
 return numberPage;  
 }  
  
 *// Метод для изменения содержания журнала* public void setNotes(Note\_[] notes) {  
 this.notes = notes;  
 }  
  
 */\*\*  
 \* Метод, формирующий информацию о журнале  
 \* @return str - возвращает строку  
 \*/* @Override  
 public String toString() {  
 StringBuilder str = new StringBuilder("Журнал " + magazineTitle + " № " +  
 numberMagazine + " " + publishingYear + ". ‐ "  
 + publishingLocation + ".: " + publishingHouse);  
  
 for (Note\_ note : notes) {  
 if (note != null) {  
 str.append("\n").append(note.toString());  
 }  
 }  
  
 return str.toString();  
 }  
}  
  
*/\*\*  
 \* Класс для представления статьи  
 \*/*class Note {  
 private String author;  
 private String title;  
 private double pages;  
  
 public Note(String author, String title, double pages) {  
 this.author = author;  
 this.title = title;  
 this.pages = pages;  
 }  
  
 @Override  
 public String toString() {  
 return author + ". " + title + ", авт. листов: " + pages;  
 }  
}

**Скриншот результата:**

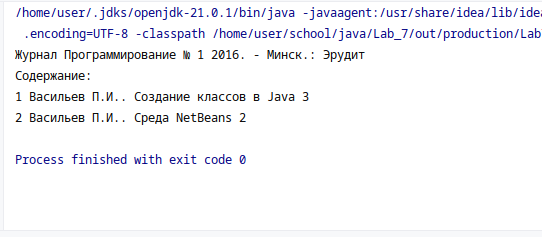


Рисунок 1 – результат работы программы AggregationExample

**Задание 2:**

Изучите и законспектируйте теорию:

1. Композиция

2. Наследование

Самостоятельно:

Реализовать код приведенный в разделе про агрегацию

Написать javadoc комментарий к классу и методам.

**Код программы:**

*/\*\*  
 \* Проверка классов  
 \*  
\* @author Tsimoshchenko M.V, Kudryavets D.E  
\* @version 1.0  
 \*/*public class CompositionExample {  
 public static void main(String[] args) {  
 Note\_ noteVasilev1 = new Note\_("Васильев П.И.", "Создание классов в Java", 3);  
 Note\_ noteVasilev2 = new Note\_("Васильев П.И.", "Среда NetBeans", 42);  
  
 Note\_[] notes = new Note\_[10];  
 notes[0] = noteVasilev1;  
 notes[1] = noteVasilev2;  
  
 String PUBLISHING\_HOUSE = "Эрудит";  
 String PUBLISHING\_LOCATION = "Минск";  
  
 Magaz magazine = new Magaz("Программирование", 1, 2016,  
 notes, PUBLISHING\_HOUSE,  
 PUBLISHING\_LOCATION);  
  
 *// Вывод информации о журнале* System.*out*.println(magazine.toString());  
  
 *// Вывод общего количества страниц* System.*out*.println("Всего страниц: " + magazine.getNumberPage());  
 }  
}  
  
class Magaz {  
 private String magazineTitle;  
 private int numberMagazine;  
 private int publishingYear;  
 private Note\_[] notes;  
 private String publishingHouse;  
 private String publishingLocation;  
  
 public Magaz(String magazineTitle, int numberMagazine,  
 int publishingYear, Note\_[] notes,  
 String publishingHouse, String publishingLocation) {  
 this.magazineTitle = magazineTitle;  
 this.numberMagazine = numberMagazine;  
 this.publishingYear = publishingYear;  
 this.notes = notes;  
 this.publishingHouse = publishingHouse;  
 this.publishingLocation = publishingLocation;  
 }  
  
 public String getMagazineTitle() {  
 return magazineTitle;  
 }  
  
 public int getNumberMagazine() {  
 return numberMagazine;  
 }  
  
 public int getPublishingYear() {  
 return publishingYear;  
 }  
  
 public Note\_[] getNotes() {  
 return notes;  
 }  
  
 public String getPublishingHouse() {  
 return publishingHouse;  
 }  
  
 public String getPublishingLocation() {  
 return publishingLocation;  
 }  
  
 *// Метод для подсчета общего количества страниц* public int getNumberPage() {  
 int totalPages = 0;  
 for (Note\_ note : notes) {  
 if (note != null) {  
 totalPages += note.getPages();  
 }  
 }  
 return totalPages;  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder str = new StringBuilder("Журнал " + magazineTitle + " № " +  
 numberMagazine + " " + publishingYear + ". ‐ "  
 + publishingLocation + ".: " + publishingHouse + "\nСодержание:");  
  
 int index = 1; *// Индекс для нумерации статей* for (Note\_ note : notes) {  
 if (note != null) {  
 str.append("\n").append(index++).append(" ").append(note.toString());  
 }  
 }  
  
 return str.toString();  
 }  
}  
  
class Note\_ {  
 private String author;  
 private String title;  
 private double pages;  
  
 public Note\_(String author, String title, double pages) {  
 this.author = author;  
 this.title = title;  
 this.pages = pages;  
 }  
  
 public double getPages() {  
 return pages; *// Геттер для количества страниц* }  
  
 @Override  
 public String toString() {  
 return author + ". " + title + " " + (int)pages; *// Приводим к целому числу для вывода* }  
}

**Скриншот результата:**

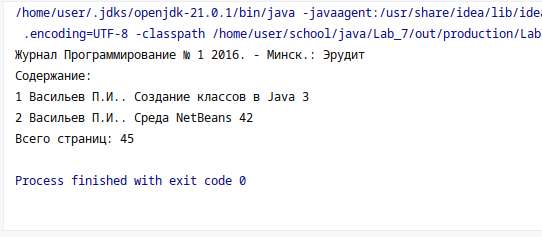


Рисунок 2 – результат работы программы СompositionExample

**Контрольные вопросы**

**1. Что такое конструктор по умолчанию?**

Конструктор по умолчанию — это специальный метод в классе, который вызывается при создании нового объекта этого класса. Если вы не определяете ни одного конструктора в классе, Java автоматически предоставляет конструктор по умолчанию, который не принимает параметров и инициализирует поля класса значениями по умолчанию (например, 0 для чисел, null для объектов и false для булевых значений).

Если вы определяете хотя бы один конструктор с параметрами, конструктор по умолчанию не будет автоматически создан. В этом случае, если вам все же нужен конструктор без параметров, вы должны определить его явно.

▎Пример конструктора по умолчанию

class MyClass {

int value;

// Конструктор по умолчанию

MyClass() {

value = 10; // Инициализация поля

}

}

// Использование

public class Main {

public static void main(String[] args) {

MyClass instance = new MyClass(); // Вызов конструктора по умолчанию

System.out.println(instance.value); // Вывод: 10

}

}

**2.** **Какие модификаторы доступа рекомендуется использовать для полей,**

**методов?**

В Java существуют четыре основных модификатора доступа:

1. public: Поля или методы с этим модификатором доступны из любого другого класса. Рекомендуется использовать для методов, которые должны быть доступны извне.

2. private: Поля или методы с этим модификатором доступны только внутри самого класса. Рекомендуется использовать для полей, чтобы защитить их от прямого изменения извне. Это помогает поддерживать инкапсуляцию.

3. protected: Поля или методы с этим модификатором доступны в классе, его подклассах и классах в одном пакете. Обычно используется, когда нужно предоставить доступ к полям или методам для наследников.

4. default (package-private): Если не указан ни один модификатор, считается, что это "пакетный" доступ. Поля или методы доступны только в рамках одного пакета. Это может быть полезно для организации кода внутри пакета.

▎Рекомендации по использованию модификаторов доступа

- Поля: Рекомендуется использовать private для полей класса, чтобы защитить их от нежелательного доступа и изменений извне. Доступ к ним следует предоставлять через геттеры и сеттеры.

- Методы: Используйте public для методов, которые должны быть доступны извне. Для вспомогательных методов, которые используются только внутри класса, используйте private. Если метод должен быть доступен в подклассах, используйте protected.

**Вывод**

Изучил наследование в языке Java, использование агрегации и композиции.