МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

по дисциплине «Визуальные средства разработки программных приложений»

на тему: «Использование внутренних классов*»*

Выполнил: студент гр. ИТП-31

Прошко Д.А.

Принял: преподователь

Ястребов А. А.

Гомель 2024

**Цель работы:** Изучить методы создания моделей программных систем и реализации их с использованием языка программирования *Java.*

**Задание:** Создать приложение, удовлетворяющее требованиям, приведённым в задании. Аргументировать принадлежность классу каждого создаваемого метода.

1. Для определения иерархии классов связать отношением наследования классы, приведенные в приложении (для заданного варианта). Из перечисленных классов выбрать один, который будет стоять во главе иерархии. Это абстрактный класс.

2. Определить в классах все необходимые конструкторы.

3. Компонентные данные класса специфицировать как *protected*.

4. Для добавления объекта в список предусмотреть метод классаа, т.е. объект сам добавляет себя в список. Например, *a.Add()* *a* добавляет себя в список.

5. Включение объекта в список можно выполнять при создании объекта, т.е. поместить операторы включения в конструктор. В случае иерархии классов, включение объекта в список должен выполнять только конструктор базового класса. Вы должны продемонстрировать оба этих способа.

6. Список просматривать путем вызова метода *Show* каждого объекта.

7. Статический метод просмотра списка вызывать не через объект, а через класс.

**Вариант:**

Перечень классов:

6) журнал, книга, печатное издание, учебник.

**Ход работы**

Обоснование принадлежности методов к классам:

* Конструкторы:

В каждом классе определяется конструктор для инициализации данных и добавления объекта в список. Конструктор базового класса *Place* добавляет объект в общий список.

* Метод *add*:

Этот метод добавляет объект в список. Он определен в классе *Place*, чтобы все наследники могли его использовать.

* Метод *show*:

Абстрактный метод show определен в *Place* и реализуется в каждом наследуемом классе для вывода специфичной информации о конкретном месте.

* Статический метод *showAll*:

Метод определен в классе *Publication* и предоставляет возможность просмотра всех объектов, добавленных в список. Вызывается без создания экземпляра класса.

*UML*-схема классов представлена на рисунке 1. Результат выполнения задания представлен на рисунке 2.

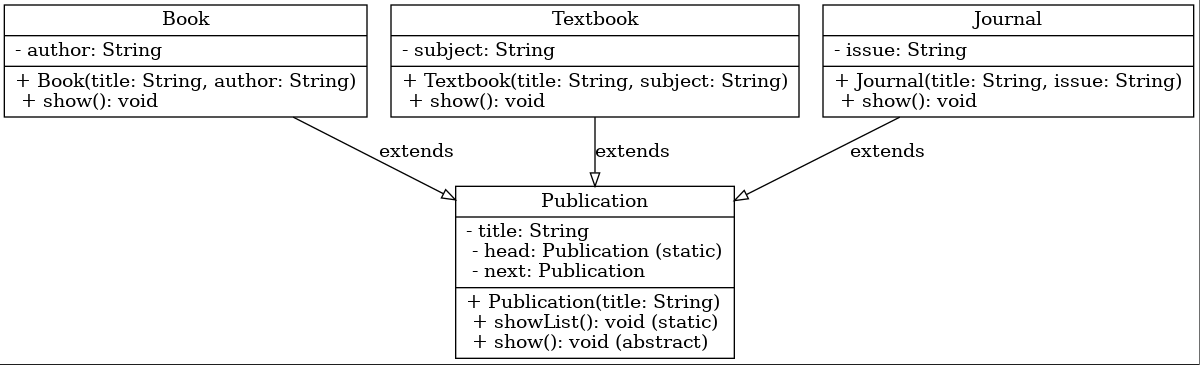


Рисунок 1 – *UML*-схема классов

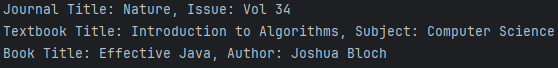


Рисунок 2 – Результат выполнения задания

**Вывод:** В результате выполнения лабораторной работы были изучены методы создания моделей программных систем и реализации их с использованием языка программирования *Java*, а также была разработана программа, реализующая иерархию классов.

**ПРИЛОЖЕНИЕ А**

**Текст программы**

**Main:**

package publication;

import publication.items.Book;

import publication.items.Textbook;

import publication.items.Journal;

public class Main {

public static void main(String[] args) {

new Book("Effective Java", "Joshua Bloch");

new Textbook("Introduction to Algorithms", "Computer Science");

new Journal("Nature", "Vol 34");

Publication.showList();

}

}

**Publication:**

package publication;

public abstract class Publication {

protected String title;

protected static Publication head;

protected Publication next;

public Publication(String title) {

this.title = title;

this.next = head;

head = this;

}

public static void showList() {

Publication current = head;

while (current != null) {

current.show();

current = current.next;

}

}

public abstract void show();

}

**Bookв:**

package publication.items;

import publication.Publication;

public class Book extends Publication {

private String author;

public Book(String title, String author) {

super(title);

this.author = author;

}

@Override

public void show() {

System.out.println("Book Title: " + title + ", Author: " + author);

}

}

**Textbook:**

package publication.items;

import publication.Publication;

public class Textbook extends Publication {

private String subject;

public Textbook(String title, String subject) {

super(title);

this.subject = subject;

}

@Override

public void show() {

System.out.println("Textbook Title: " + title + ", Subject: " + subject);

}

}

**Journal:**

package publication.items;

import publication.Publication;

public class Journal extends Publication {

private String issue;

public Journal(String title, String issue) {

super(title);

this.issue = issue;

}

@Override

public void show() {

System.out.println("Journal Title: " + title + ", Issue: " + issue);

}

}