**Лабораторна робота №3**

**Розробка та реалізація програм із використанням класів для опису даних та створення масивів об’єктів**

**Ціль роботи**: отримати навички створення та реалізації програм з використанням масивів об’єктів та вивчити методи роботи зі строковими даними.

**Завдання**

1. Створити проект, що складається з двох класів: основного (Main) та класу для представлення об’єкта відповідно специфікації, що наведена у таблиці 1. Кожний клас повинен бути розміщений у окремому пакеті. У створеному класі визначити приватні поля для зберігання указаних даних, конструктори для створення об’єктів та відкриті методи setValue(), getValue(), toString() для доступу до полів об’єкту.

2. В основному класі програми визначити методи, що створюють масив об'єктів. Задати критерії вибору даних та вивести ці дані на консоль. Для кожного критерію створити окремий метод.

3. Виконати програму, та пересвідчитись, що дані зберігаються та коректно виводяться на екран відповідно до вказаних критеріїв.

4. Створити ще один проект, що складається з двох класів: основного (Main) та класу для завдання згідно варіанту таблиці 2.

Примітка:

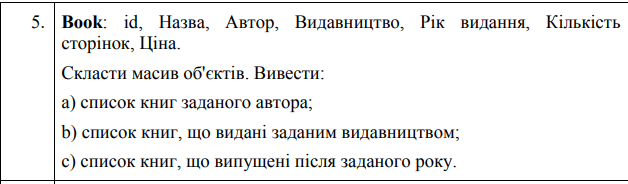
1.а) В усіх варіантах завдань вважати, що текст складається з літер та пропусків. Словом будемо називати непорожню послідовність літер, яка не містить пропусків.

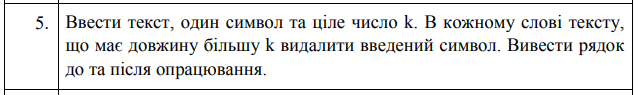
1.b) Додаткове завдання (для всіх варіантів)

Виконати теж саме завдання, вважаючи, що слова у текстах можуть містити лише літери, але відокремлюються не тільки пропусками, а й іншими символами, що не є літерами і цифрами (наприклад, знаки пунктуації).

2. Виконати програму, та пересвідчитись, що дані зберігаються та коректно

виводяться на екран відповідно до вказаних критеріїв.





**Короткі теоретичні відомості**

Масиви:

Типи масиву використовуються для визначення масивів – упорядкованих наборів однотипних змінних. Ви можете визначити масив над будь-яким існуючим у мові типом, включаючи типи, визначені користувачем. Крім того, можна користатися масивами масивів чи багатовимірними масивами. Коротко говорячи, якщо ми можемо створити змінну деякого типу, виходить, ми можемо створити і масив змінних цього типу. Разом з тим створення масивів у мові Java може показатися вам незвичним, тому що воно вимагає застосування оператора new.

Клас String

Кожен рядок, створюваний за допомогою оператора new або за допомогою літерала (укладений в подвійні апострофи), є об'єктом класу String. Особливістю об'єкта класу String є те, що його значення не може бути змінено після створення об'єкту. При виклику будь-якого методу класу, результатом може бути новий рядок, але не змінюється початковий. Таким чином будь-який метод, що намагається змінити рядок приводить до створення нового об'єкта.

Класи StringBuilder та StringBuffer є “близнюками” та за призначенням наближені до класу String, але, на відмінність від останнього, вміст та розміри об'єктів класів StringBuilder та StringBuffer можна змінювати. Тому, якщо потрібно виконувати багато операцій з перетворення рядків, треба в першу чергу, розглядати можливість використання саме об’єктів класів StringBuilder та StringBuffer. За допомогою відповідних методів та конструкторів об'єкти класів StringBuffer, StringBuilder та String можна перетворювати один до одного. Конструктор класу StringBuffer (так саме як і StringBuilder) може приймати в якості параметра об'єкт String або невід'ємний розмір буфера. Об'єкти цього класу можна перетворювати в об'єкт класу String методом toString() або за допомогою конструктора класу String.

*Розв’язання*

**Частина 1:**

1. **Текст програми**

Lab3Part1.java:

**package** com.labs.lab3.part1;  
  
**import** com.labs.lab3.part1.library.\*;  
**import static** com.helpers.console.ConsolePrompt.\*;  
**import static** com.helpers.console.ConsoleElements.*hr*;  
  
**public class** Lab3Part1 {  
 **private** Library **library** = **new** Library();  
  
 **public static void** main(String[] args) {  
  
 }  
  
 **public void** run() {  
 Book[] books;  
  
 **var** type = *promptInt*(  
 **"Enter \"1\" if you want to enter books yourself, "** + **"something else to use a test books: "**);  
  
 **if** (type == 1)  
 books = getBooksFromConsole();  
 **else** books = getTestBooks();  
  
 **library**.addBooks(books);  
 runCLI();  
 }  
  
 **private void** runCLI() {  
 printHelp();  
  
 **while** (**true**) {  
 String command = *promptLine*(**"Enter command: "**);  
  
 **if** (command.equalsIgnoreCase(**"exit"**))  
 **break**;  
 **else** useCommand(command);  
  
 *hr*();  
 }  
  
 }  
  
 **private void** useCommand(String command) {  
 **if** (command.equalsIgnoreCase(**"author"**)) {  
 String author = *promptLine*(**"Enter the author: "**);  
 printBooks(**library**.getBooksByAuthor(author));  
  
 } **else if** (command.equalsIgnoreCase(**"publisher"**)) {  
 String publisher = *promptLine*(**"Enter the publisher: "**);  
 printBooks(**library**.getBooksByPublisher(publisher));  
  
 } **else if** (command.equalsIgnoreCase(**"afterYear"**)) {  
 **int** year = *promptInt*(**"Enter the year: "**);  
 printBooks(**library**.getBooksAfterYear(year));  
  
 }  
  
 **else if** (command.equalsIgnoreCase(**"all"**)) {  
 printBooks(**library**.getBooks());  
  
 } **else if** (command.equalsIgnoreCase(**"help"**)) {  
 printHelp();  
  
 } **else** {  
 System.***out***.println(**"Command not found. Try again: "**);  
 }  
 }  
  
 **private** Book[] getBooksFromConsole() {  
 **var** number = *promptInt*(**"Enter the number of books: "**);  
 Book[] books = **new** Book[number];  
  
 *hr*();  
  
 **for** (**int** i = 0; i < books.**length**; i++) {  
 String name = *promptLine*(**"Enter the name of the book: "**);  
 String author = *promptLine*(**"Enter the author: "**);  
 String publisher = *promptLine*(**"Enter the publisher: "**);  
 **int** year = *promptInt*(**"Enter the year: "**);  
 **int** pages = *promptInt*(**"Enter the number of pages: "**);  
 **double** price = *promptDouble*(**"Enter the price: "**);  
  
 books[i] = **new** Book(name, author, publisher, year, pages, price);  
  
 *hr*();  
 }  
  
 **return** books;  
 }  
  
 **private** Book[] getTestBooks() {  
 Book[] books = **new** Book[7];  
  
 books[0] = **new** Book(**"JavaScript: The Definitive Guide"**, **"David Flanagan"**, **"O'Reilly"**, 2006, 999, 50);  
 books[1] = **new** Book(**"Java Programming"**, **"Donald Bales"**, **"O'Reilly"**, 2001, 450, 60);  
 books[2] = **new** Book(**"Thinking in Java"**, **"Bruce Eckel"**, **"Oracle"**, 2002, 328, 50);  
 books[3] = **new** Book(**"JS.Next"**, **"Aaron Frost"**, **"O'Reilly"**, 2015, 250, 50);  
 books[4] = **new** Book(**"You don't know JS"**, **"Simpson K."**, **"O'Reilly"**, 2015, 88, 30);  
 books[5] = **new** Book(**"Angular"**, **"David Flanagan"**, **"O'Reilly"**, 2016, 396, 30);  
 books[6] = **new** Book(**"Java in a Nutshell"**, **"David Flanagan"**, **"O'Reilly"**, 2014, 396, 60);  
  
 **return** books;  
 }  
  
 **private void** printBooks(Book[] books) {  
 **if** (books.**length** == 0) {  
 System.***out***.println(**"There's no book"**);  
 }  
  
 **for** (Book book : books) {  
 System.***out***.println(book);  
 }  
 }  
  
 **private void** printHelp() {  
 System.***out***.println(**"Commands:"**);  
 System.***out***.println(**"> all #Print all books"**);  
 System.***out***.println(**"> author #Print books by Author"**);  
 System.***out***.println(**"> publisher #Print books by Publisher"**);  
 System.***out***.println(**"> afterYear #Print books published after year"**);  
 System.***out***.println();  
 System.***out***.println(**"> help #Print Help"**);  
 System.***out***.println(**"> exit #Exit"**);  
  
 *hr*();  
 }  
}

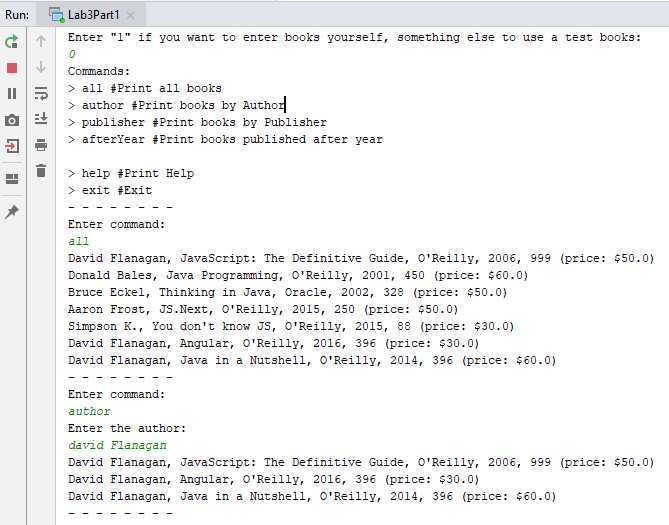
Library.java

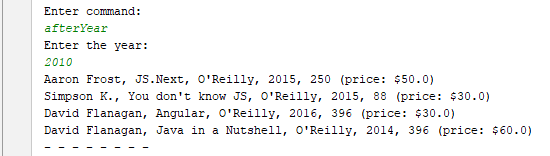
**package** com.labs.lab3.part1.library;  
  
**import** java.util.Arrays;  
  
**public class** Library {  
 **private** Book[] **books**;  
  
 **public** Book[] getBooks() {  
 **return** Arrays.*copyOf*(**books**, **books**.**length**);  
 }  
  
 */\*\*  
 \* Добавляет переданные книги  
 \** ***@param books*** *Книги, что нужно добавить  
 \*/* **public void** addBooks(Book[] books) {  
 **this**.**books** = books;  
  
 Book[] newBooks = Arrays.*copyOf*(**this**.**books**, **this**.**books**.**length** + books.**length**);  
 System.*arraycopy*(books, 0, newBooks, **this**.**books**.**length**, books.**length**);  
 }  
  
 */\*\*  
 \* Взвращает массив книг переданного автора  
 \** ***@param author*** *Автор  
 \** ***@return*** *массив книг переданного автора  
 \*/* **public** Book[] getBooksByAuthor(String author) {  
 **return** Arrays.*stream*(**books**)  
 .filter(book -> book.getAuthor().equalsIgnoreCase(author))  
 .toArray(Book[]::**new**);  
 }  
  */\*\*  
 \* Взвращает массив книг переданного издательства  
 \** ***@param publisher*** *Издательство  
 \** ***@return*** *массив книг переданного издательства  
 \*/* **public** Book[] getBooksByPublisher(String publisher) {  
 **return** Arrays.*stream*(**books**)  
 .filter(book -> book.getPublisher().equalsIgnoreCase(publisher))  
 .toArray(Book[]::**new**);  
 }  
  
 */\*\*  
 \* Взвращает массив книг изданных после переданного года  
 \** ***@param year*** *Год  
 \** ***@return*** *массив книг изданных после переданного года  
 \*/* **public** Book[] getBooksAfterYear(**int** year) {  
 **return** Arrays.*stream*(**books**)  
 .filter(book -> book.getYear() > year)  
 .toArray(Book[]::**new**);  
 }  
  
}

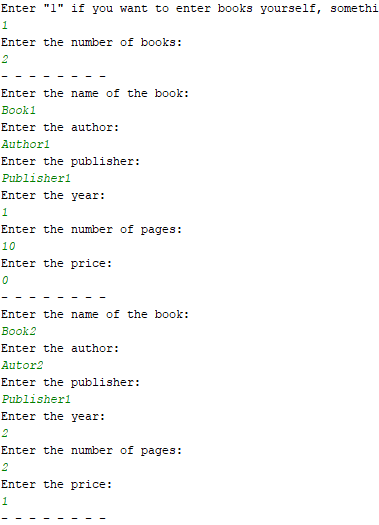
Book.java

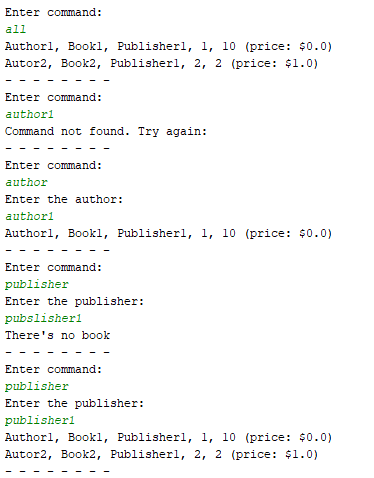
**package** com.labs.lab3.part1.library;  
  
**import** java.time.LocalDate;  
**import** java.util.UUID;  
  
**public class** Book {  
 **private final** UUID **id** = UUID.*randomUUID*();  
 **private** String **name** = **"No name"**;  
 **private** String **author** = **"No author"**;  
 **private** String **publisher** = **"No publisher"**;  
 **private** LocalDate **date** = LocalDate.*of*(0,1,1);  
 **private int pages** = 0;  
 **private double price** = 0;  
  
 **public** Book() {  
  
 }  
  
 **public** Book(String name, String author, String publisher, **int** year, **int** pages, **double** price) {  
 setName(name);  
 setAuthor(author);  
 setPublisher(publisher);  
 setYear(year);  
 setPages(pages);  
 setPrice(price);  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public** String getAuthor() {  
 **return author**;  
 }  
  
 **public void** setAuthor(String author) {  
 **this**.**author** = author;  
 }  
  
 **public** String getPublisher() {  
 **return publisher**;  
 }  
  
 **public void** setPublisher(String publisher) {  
 **this**.**publisher** = publisher;  
 }  
  
 **public int** getPages() {  
 **return pages**;  
 }  
  
 **public void** setPages(**int** pages) {  
 **this**.**pages** = pages;  
 }  
  
 **public double** getPrice() {  
 **return price**;  
 }  
  
 **public void** setPrice(**double** price) {  
 **this**.**price** = price;  
 }  
  
 **public int** getYear() {  
 **return date**.getYear();  
 }  
  
 **public void** setYear(**int** year) {  
 **date** = LocalDate.*of*(year, **date**.getMonth(), date.getDayOfYear());  
 }  
  
 @Override  
 **public** String toString() {  
 **return** getAuthor() + **", "** +  
 getName() + **", "** +  
 getPublisher() + **", "** +  
 getYear() + **", "** +  
 getPages() + **" (price: $"** +  
 getPrice() + **")"**;  
 }  
}

1. Результат виконання програми:









**Частина 2**

1. Текст програми

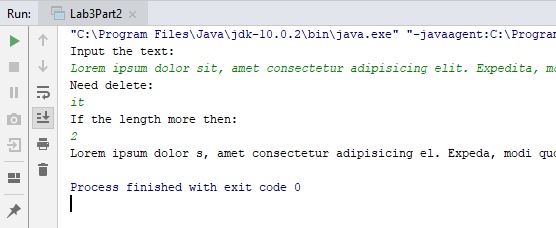
***Lab3Part2.java***

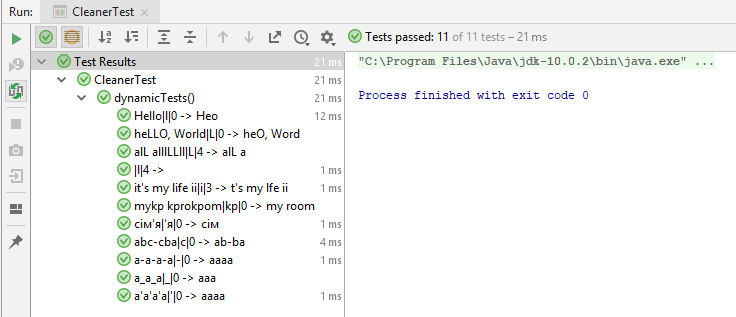
**package** com.labs.lab3.part2;  
  
**import static** com.helpers.console.ConsolePrompt.*promptInt*;  
**import static** com.helpers.console.ConsolePrompt.*promptLine*;  
  
**public class** Lab3Part2 {  
 **public static void** main(String[] args) {  
 **var** app = **new** Lab3Part2();  
 app.run();  
 }  
  
 **public void** run() {  
 String text = *promptLine*(**"Input the text: "**);  
 String sub = *promptLine*(**"Need delete: "**);  
 **int** moreThen = *promptInt*(**"If the length more then: "**);  
  
 String cleaned = **new** Cleaner().deleteFromWords(text, sub, moreThen - 1);  
 System.***out***.println(cleaned);  
 }  
}

***Cleaner.java***

**package** com.labs.lab3.part2;  
  
**import** java.util.regex.Pattern;  
  
**public class** Cleaner {  
 **private static** Pattern *WORD\_PATTERN* = Pattern.*compile*(**"(?i)[a-zа-яёіїє\\d\\-\_']+"**);  
  
 *//Символы слова, которые не входят в количетсво букв* **private static final** String ***NO\_COUNT*** = **"[\\-\_']+"**;  
  
 */\*\*  
 \* Удалает из слов текста text длинной от minLen все вхождения подстроки delete  
 \** ***@param text*** *Исходный текст  
 \** ***@param delete*** *Подстрока, которую нужно удалить из слов  
 \** ***@param minLen*** *Минимальная длина слова  
 \** ***@return*** *строка образованная из входной путем удаления в словах указанной подстроки  
 \*/* **public** String deleteFromWords(String text, String delete, **int** minLen) {  
 **final** String deletePattern = **"(?i)"** + delete;  
  
 **return** *WORD\_PATTERN*.matcher(text).replaceAll((matchResult) -> {  
 String word = matchResult.group();  
 String cleanWord = word.replaceAll(***NO\_COUNT***, **""**);  
  
 **if** (cleanWord.length() >= minLen) {  
 **return** word.replaceAll(deletePattern, **""**);  
 }  
  
 **return** word;  
 });  
 }  
  
}

1. Результат виконання програми:





**Висновок:** на цій лабораторній роботі оволодів основами методології **TDD** та навичками роботи з алгоритмічними структурами мови **Java.**