УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Кафедра інформаційних технологій та програмної інженерії

КУРСОВА РОБОТА

з Об’єктно-орієнтованого програмування

на тему: «Класовий тип для обробки текстової інформації»

Студента II курсу____групи
галузі знань 12 «Інформаційні технології»
спеціальності 121 «Інженерія програмного забезпечення»
Бурлака В.С.
(прізвище та ініціали)

Керівник _____к.т.н., Пашкевич О.П.

Національна шкала _____

Кількість балів _____ Оцінка: ECTS _____

м. Івано-Франківськ

2021

Університет Короля Данила

назва вищого навчального закладу

Кафедра інформаційних технологій та програмної інженерії

Дисципліна Об'єктоно-орієнтоване програмування

Спеціальність 121 «Інженерія програмного забезпечення»

Курс II Група ІПЗс-2019 Семестр 4

ЗАВДАННЯ

на курсовий проект (роботу) студента

Бурлака Владислав Сергійович

Прізвище, ім'я, по-батькові

1. Тема проекту (роботи) Класовий тип для обробки текстової інформації

2. Строк здачі студентом закінченого проекту (роботи) _____

3. Вихідні дані до проекту (роботи)

4. Зміст роботи (перелік питань по розділах, які потрібно розробити)

5. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсового проекту (роботи)	Строк виконання	Примітки
1	Ознайомлення з завданням	3-4.05	
2	Створення класів	6-7.05	
3	Розробка методів	10.05	
4	Перша половина коду	12-15.05	
5	Друга половина коду	15-18.05	
6	Оформлення курсової роботи	20-23.05	

Студент _____ _____

Підпис Прізвище, ініціали

Керівник _____ _____

Підпис Прізвище, ініціали

«____» _____ 2021__ р.

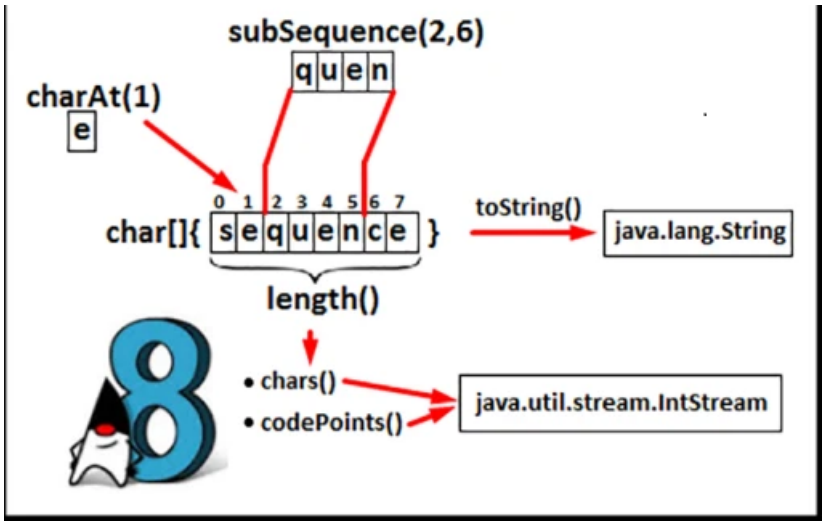
ЗМІСТ

ВСТУП

Java є мовою програмування і платформу обчислень, яка була вперше випущена Sun Microsystems в 1995 р Існує безліч додатків і веб-сайтів, які не працюють при відсутності встановленої Java, і з кожним днем число таких веб-сайтів і додатків збільшується. Java відрізняється швидкістю, високим рівнем захисту і надійністю. Від портативних комп'ютерів до центрів даних, від

ігрових консолей до комп'ютерів, які використовуються для наукових розробок, від стільникових телефонів до мережі Інтернет - Java всюди!

Власне, **String** в перекладі з англійської - рядок. Так і є, тип String представляє текстовий рядок. А чим же є текстовий рядок? Текстовий рядок - це якась впорядкована послідовність символів, які йдуть один за одним. Символ - char. Послідовність - sequence. Так що так, абсолютно правильно, String є реалізацією java.lang.CharSequence. А якщо заглянути всередину самого класу String, то всередині нього ніщо інше як масив char'ов: private final char value[]; У java.lang.CharSequenceдосить простий контракт:



У нас є метод отримання кількості елементів, отримання конкретного елемента і отримання набору елементів + безпосередньо сам метод toString, який поверне this) Цікавіше розібратися в методах, які прийшли до нас в Java 8, а це: chars()і codePoints().

Згадуємо по Tutorial від Oracle « Primitive Data Types », що char - це single 16-bit Unicode character.Тобто по суті char це просто тип розміром в половину типу int (32 біта), який представляє числа від 0 до 65535 (див. decimal значення в ASCII Table). Тобто при бажанні ми можемо char представити у вигляді int. І в Java 8 цим скористалися. Починаючи з 8 версії Java у нас з'являється **IntStream**- стрім для роботи з примітивними int'амі. Тому в charSequence є можливість отримати IntStream, що представляє або char'и або codePoint'и.

Моє завдання полягає в розробленні демонстраційно-тестуючої програми, яка міститиме у собі класовий тип для обробки текстової інформації String. А також створити і реалізувати компонентні методи які знаходяться в “Таблиці 1.

Таблиця 1 – Методи, які необхідно реалізувати в програмі.

Назва методу	Реалізація
Length()	довжина рядка;
Substring()	пошук субрядку;
SubstringEdit()	пошук субрядку та його заміна;
ToUpperCase()	приведення усіх букв до верхнього регістру;
ToLowerCase()	приведення усіх букв до нижнього регістру;
Concate()	конкатенація;

ЗАГАЛЬНЕ ПРО ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Об'єктно-орієнтоване програмування (ООП) *- одна з парадигм програмування, яка розглядає програму як множину "об'єктів", що взаємодіють між собою*. В ній використано декілька технологій від попередніх парадигм, зокрема успадкування, модульність, поліморфізм та інкапсуляцію. Незважаючи на те, що ця парадигма з'явилась в 1960-тих роках, вона не мала широкого застосування до 1990-тих.

Об'єктно-орієнтовані мови програмування:

- класичні: Simula 67, Smalltalk, Actor, Eiffel, Objective C;
- сучасні: C++, Object Pascal, Java, C#.

Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування Симула в 1960-тих роках, одночасно з посиленням дискусій про кризу програмного забезпечення. Разом із тим, як ускладнювалось апаратне та програмне забезпечення, було дуже важко зберегти якість програм. Об'єктно-орієнтоване програмування частково розв'язує цю проблему шляхом наголошення на модульності програми.

На відміну від традиційних поглядів, коли програму розглядали як набір підпрограм або як перелік інструкцій комп'ютеру, ООП програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожний об'єкт здатний отримувати повідомлення, обробляти дані та надсилати повідомлення іншим об'єктам. Кожен об'єкт - своєрідний незалежний автомат з окремим призначенням та відповідальністю.

Об'єктно-орієнтоване програмування - це метод програмування, оснований на поданні програми у вигляді сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування.

На думку Алана Кея, розробника мови Smalltalk, якого вважають одним з "батьків-засновників" ООП, об'єктно-орієнтований підхід полягає в наступному наборі **основних принципів**:

• Все є об’єктами.

• Всі дії та розрахунки виконуються шляхом взаємодії (обміну даними) між об’єктами, при якій один об’єкт потребує, щоб інший об’єкт виконав деяку дію. Об’єкти взаємодіють, надсилаючи і отримуючи повідомлення. Повідомлення - це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися при виконанні дії.

• Кожен об’єкт має незалежну пам’ять, яка складається з інших об’єктів.

• Кожен об’єкт є представником (екземпляром, примірником) класу, який виражає загальні властивості об’єктів.

• У класі задається поведінка (функціональність) об’єкта. Таким чином усі об’єкти, які є екземплярами одного класу, можуть виконувати одні й ті ж самі дії.

• Класи організовані у єдину деревовидну структуру з загальним корінням, яка називається ієрархією успадкування. Пам’ять та поведінка, зв’язані з екземплярами деякого класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

Таким чином, програма являє собою набір об’єктів, що мають стан та поведінку. Об’єкти взаємодіють використовуючи повідомлення. Будується ієрархія об’єктів: програма в цілому — це об’єкт, для виконання своїх функцій вона звертається до об’єктів що містяться у ньому, які у свою чергу виконують запит шляхом звернення до інших об’єктів програми. Звісно, щоб уникнути безкінечної рекурсії у зверненнях, на якомусь етапі об’єкт трансформує запит у повідомлення до стандартних системних об’єктів, що даються мовою та середовищем програмування. Стійкість та керованість системи забезпечуються за рахунок чіткого розподілення відповідальності об’єктів (за кожну дію відповідає певний об’єкт), однозначного означення інтерфейсів міжоб’єктної взаємодії та повної ізольованості внутрішньої структури об’єкта від зовнішнього середовища (інкапсуляції).

ПРИНЦИПИ ООП

головне

- Інкапсулюють все, що може змінюватися;
- Приділяйте більше уваги інтерфейсів, а не їх реалізацій;
- Кожен клас в вашому додатку повинен мати тільки одне призначення;
- Класи - це їхня поведінка і функціональність.

Базові принципи ООП

- Абстракція - відділення концепції від її примірника;
- Поліморфізм - реалізація завдань однієї і тієї ж ідеї різними способами;
- Спадкування - здатність об'єкта або класу базуватися на іншому об'єкті або класі. Це головний механізм для повторного використання коду. Спадкове ставлення класів чітко визначає їх ієрархію;
- Інкапсуляція - розміщення одного об'єкта або класу всередині іншого для розмежування доступу до них.

Використовуйте наступне разом з успадкуванням

- Делегація - передоручення завдання від зовнішнього об'єкта внутрішнього;
- Композиція - включення об'єктом-контейнером об'єкта-та обробка його поведінкою; останній не може існувати поза першого;
- Агрегація - включення об'єктом-контейнером посилання на об'єкт-вміст; при знищенні першого останній продовжує існування.

Чи не повторюється (Do not repeat yourself - DRY)

Уникайте повторного написання коду, виносячи в абстракції часто використовувані завдання і дані. Кожна частина вашого коду або інформації повинна знаходитися в однині в єдиному доступному місці. Це один з принципів читаного коду .

Принцип єдиною обов'язки

Для кожного класу має бути визначено єдине призначення. Всі ресурси, необхідні для його здійснення, повинні бути вміщені в цей клас і підпорядковані тільки цьому завданню.

Принцип відкритості / закритості

Програмні суті повинні бути *відкриті* для розширення, але *закриті* для змін.

Принцип підстановки лісков

Методи, які використовують якийсь тип, повинні мати можливість використовувати його підтипи, не знаючи про це.

Принцип поділу інтерфейсів

Переважно розділяти інтерфейси на більш дрібні тематичні, щоб реалізують їх класи не були змушені визначати методи, які безпосередньо в них не використовуються.

Принцип інверсії залежностей

Система повинна конструюватися на основі абстракцій «зверху вниз»: чи не абстракції повинні формуватися на основі деталей, а деталі повинні формуватися на основі абстракцій.

ЗАГАЛЬНЕ ПРО МОВУ ПРОГРАМУВАННЯ-JAVA

Java — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Java вплинула на розвиток J++[en], що розроблялась компанією «Microsoft». Роботу над J++ було зупинено через судовий позов «Sun Microsystems», оскільки ця мова програмування була модифікацією Java. Пізніше в новій платформі «Microsoft» .NET випустили J#, щоб полегшити міграцію програмістів J++ або Java на нову платформу. З часом нова мова програмування C# стала основною мовою платформи, перейнявши багато чого з Java. J# востаннє включався в версію Microsoft Visual Studio 2005. Мова сценаріїв JavaScript має схожу із Java назву і синтаксис, але не пов'язана із Java.

ПЕРЕВАГИ

Перевага 1

Java легко вивчається новачками

Синтаксис цієї мови програмування схожий на звичайну англійську мову. В ній є мінімальна кількість складних для запам’ятовування символів. По суті, після встановлення JDK, настройки PATH і вивчення особливостей Classpath спеціаліст вже може створювати елементарні програми на Java.

Перевага 2

У Java прийняті концепції хорошого програмування

Java — об’єктно-орієнтована мова, причому це саме «об’єктне» в Java реалізоване просто-напросто на відмінно! Разом з ООП Ви вивчите концепції наслідування, абстракції, поліморфізму і так далі. Java навчить концепціям, які можна застосовувати в більшості інших мов, наприклад, в Python

Перевага 3

Java така мова програмування рідко змінюється

Для новачків це ВЕЛИКИЙ плюс, що не потрібно відволікатися і постійно слідкувати за новими тенденціями IT. Новий функціонал Java інколи навіть занадто довго вводиться… наприклад, подібне відбувалося із замиканнями.

Перевага 4

Користувачі Java мають доступ до великої колекції бібліотек з відкритим кодом

Найкращі програмісти світу розробили шаблони, які роблять процедуру розробки більш простою. Підтримка з боку таких гігантів, як Google і Apache, дає зрозуміти, що ця мова програмування ще довго буде використовуватися в проектах. Java має гарно пророблений API, великий вибір інструментарію, велику кількість фреймворків.

Перевага 5

Java має гігантське співтовариство по всьому світу

Існує маса форумів, тематичних порталів та інших ресурсів, де обмінюються знаннями користувачі Java. Там спілкуються досвідчені програмісти, переймають досвід новачки, обговорюють новини, пов’язані з мовою програмування. Ви можете бути впевнені в тому, що ніяких проблем з пошуком інформації точно не буде.

Велика частина сучасних стартапів не використовує Java, так як на даний момент існують більш швидкі шляхи виконання того ж обсягу роботи. Проте ми навчаємо вивчати те, що приносить найбільше задоволення, так Ви станете справжнім експертом в обраній справі, зокрема кодуванні на улюбленій мові програмування.

Тому вивчайте мову Java. З нею ви зможете без проблем розробляти продукти, які будуть затребуваними в майбутньому. Java — це не важко. Спробуйте! І не припиняйте вивчати Java. Спробуйте також познайомитися із C ++ і Python — всі ці три мови схожі (об’єктно-орієнтовані імперативні). Вивчіть також JavaScript як слід (щоб побачити прототипну об’єктно-орієнтовану імперативний мову програмування).

ТИП char JAVA

Символи описуються в мові Java **char** типом. Символи перетворюються по таблиці кодування UTF-16. За великим рахунком це все літери, числа і спеціальні символи існують на нашій планеті.

Розмір в байтах - 2 байта

Можливі значення (от..до) - 0..65,535

Значення за замовчуванням - '\ u0000'

Тип **char** є псевдоцелочісленним типом, тому значення цього типу можна задавати у вигляді числа - коду символу з таблиці кодування UTF-16. Кожному символу відповідає певне число з таблиці і Java при вигляді цього числа в рамках типу **char** виводить його на екран як символ.

Наприклад, при виконанні цього коду

ПЕРШИЙ РОЗДІЛ

ЗАГАЛЬНО ПРО РЯДОК

Рядок або рядковий тип даних, також іноді стрічка — це тип даних, значеннями якого є довільна послідовність (рядок) символів алфавіту. Кожна змінна такого типу (рядкова змінна) може бути представлена фіксованою кількістю байтів або мати довільну довжину.

Деякі мови програмування накладають обмеження на максимальну довжину рядка, але в більшості мов подібні обмеження відсутні. При використанні Unicode кожен символ рядкового типу може вимагати двох або навіть чотирьох байтів для свого представлення.

Основні проблеми в машинному поданні рядкового типу:

- рядки можуть мати досить істотний розмір (до декількох десятків мегабайтів);
- змінюється з часом розмір — виникають труднощі з додаванням і видаленням символів.

У поданні рядків в пам'яті комп'ютера існує два принципово різних підходи.

Подання масивом символів

У цьому підході рядки представляються масивом символів; при цьому розмір масиву зберігається в окремій (службовій) області. Від назви мови Pascal, де цей метод був вперше реалізований, даний метод отримав назву Pascal strings.

Злегка оптимізованим варіантом цього методу є так званий формат c-addr-u (від англ. character-aligned address + unsigned number), застосовуваний в Форте. На відміну від Pascal strings, тут розмір масиву зберігається не спільно із рядковими даними, а є частиною покажчика на рядок.

Переваги

Програма в кожен момент часу містить відомості про розмір рядка, тому операції додавання символів в кінець, копіювання рядка і власне отримання розміру рядка виконуються досить швидко;

рядок може містити будь-які дані;

- можливо на програмному рівні стежити за виходом за межі рядка при її обробці;
- можливо швидке виконання операції виду «взяття N-го символу з кінця рядка».

РОЗРОБКА ПРОГРАМИ

Для початку створюємо клас Main, щоб вказати данний масив для операцій.

В данній програмі створенні два класи. В класі MyString будуть створенні всі методи, щоб виконувати необхідні операції з масивом:length()- знаходження довжина рядка;

Substring()- пошук субрядку; SubstringEdit()-пошук субрядку та його заміна ;ToUpperCase()-приведення усіх букв до верхнього регістру;
ToLowerCase()-приведення усіх букв до нижнього регістру;
Concate()-конкатенація;

В данній курсовій роботі ми працюємо з індексами масиву тому в рядку коду ми переводимо масив String до масиву char.

Далі в класі Main виводимо результати виконання данних методів в консоль.

ОПИС МЕТОДІВ

Для початку я створив клас Main, в ньому будуть викликатись і реалізовуватись всі методи створені в класі MyString.

```
public MyString(String str) {

    this.myStr = str.toCharArray();
    this.length = this.myStr.length;
}

public char[] getMyStr() {
    return this.myStr;
}
public Integer length() {
    return this.myStr.length;
}
```

// В данній частині коду я створюю масив і прирівнюю його до масиву чарів і повертає данний масив в консоль, а ще створений метод length ,який рахує довжину рядка.

```
public String toUpperCase() {

    this.newString = new String();
    for (int i = 0; this.myStr.length > i; i++) {
        if (Character.isUpperCase(this.myStr[i])) {
            newString += this.myStr[i];
        }
        if (Character.isLowerCase(this.myStr[i])) {
```

```
            newString += Character.toUpperCase(this.myStr[i]);
        }
        if (this.myStr[i] == ' ') {

            newString += ' ';
        }

    }
    return newString;

}
```

//Тут створення методу toUpperCase який за допомогою циклу for перебігає по всіх елементах масиву і в циклі if триває перевірка елементів масиву, якщо символ верхнього регістру, то він залишається такий як є, а відповідно якщо нижнього то переводиться у верхній регістр і повертається новий масив.

```
public String toLowerCase() {

    this.newString = new String();

    for (int i = 0; this.myStr.length > i; i++) {
        if (Character.isLowerCase(this.myStr[i])) {
            newString += this.myStr[i];
        }
        if (Character.isUpperCase(this.myStr[i])) {
            newString += Character.toLowerCase(this.myStr[i]);
        }
        if (this.myStr[i] == ' ') {
```

```
            newString += ' ';
        }
    }
    return newString;
}
```

//Тут створення методу toLowerCase який за допомогою циклу for перебігає по всіх елементах масиву і в циклі if триває перевірка елементів масиву, якщо символ нижнього регістру, то він залишається такий як є, а відповідно якщо верхнього то переводиться у нижній регістр і повертається новий масив.

```
public char[] Concate(MyString string_2) {
    int len = this.myStr.length + string_2.myStr.length;
    char[] Concate = new char[len];
    int position = 0;
    for (char object : myStr) {
        Concate[position] = object;
        position++;
    }
    for (char object : string_2.myStr) {
        Concate[position] = object;
        position++;
    }
    this.myStr = Concate;
    return Concate;
```

// В данному методі Concate створюється ще один масив, в якому за допомогою конкатенації поєднаються два масиви типу char в один і повертає його.

```
public String substring (int start, int end) {
    if (start < 0 || end > this.myStr.length) {
        return "Bad parameters";
```

```
        }
        String string = new String(this.myStr);
        return string.substring(start, end);
    }
```

// В данній частинні коду відбувається створення і реалізація методу substring, триває пошук по вказаному індексу і вивід тієї частини коду.

```
   public String substringEdit (int start, int end, String insertString) {
        String leftString = new String(this.myStr).substring(0, start);
        String rightString = new String(this.myStr).substring(end, this.myStr.length);
        return String.format("%s%s%s", leftString, insertString, rightString);
    }
```

// В методі sudstringEdit триває заміна субрядку, і його вивід.

```
public class Main {
    public static void main(String[] args) {
        MyString str = new MyString("Aloha From Here");
        MyString gtr = new MyString("Hello world");

        System.out.println(gtr.getMyStr());
        System.out.println(str.getMyStr());
        System.out.println(str.length());
        System.out.println(str.toUpperCase());
        System.out.println(str.toLowerCase());
        System.out.println(str.substring(1, 6));
        System.out.println(str.substringEdit(0, 5, "Hey"));

    }}
```

// В класі Main задані два масиви з якими я працював в методах і відбувається їх виклик для виводу в консолі.

Результат програми розміщений на малюнку нижче.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алгазинов Е.К. Анализ и компьютерное моделирование информационных процессов и систем / Е.К. Алгазинов, А.А. Сирота. – М.: Диалог МИФИ, 2009. – 414 с

2. Дейтел Х.М., Дейтел П.Дж. Как программировать на Java. Книга 1. Основы программирования – М.: Бином, 2006. – 848 с.

3. Освоюємо Java – Вікіпідручник // http://uk.wikibooks.org/wiki/Освоюємо_Java

4. Java Tutorials // http://docs.oracle.com/javase/tutorial.

5. 4.1. Робота з рядками | Java & PHP javaphp.ptngu.com.

6. Переваги мови програмування Java qagroup.com.ua.

7. Об' єктно - орієнтоване програмування - Українське програмування programming.in.ua.

8. тип данных "char" в java - CodeRoadcoderoad.ru.

9. https://www.google.com/amp/s/www.geeksforgeeks.org/java-string-concat-examples/amp/.

10. https://vertex-academy.com/tutorials/ru/substring-v-java/.

11. Шилдт Г. Полный справочнык по Java. Java SE 6 Edition, 7-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2007. – 1034 с.

12. Флэнаган Д. Java. Справочник, 4-е изд.: Пер. с англ. – Спб.:Зубов В.С. Справочник программиста. Базовые методы решения графовых задач и сортировки.- М.:"Филинъ", 1999.- 256 с.

13. Эккель Б. Философия Java.: 4-е изд. – СПб.: Питер, 2009. – 640 с.

14. Хорстманн К. С., Корнелл Г. Библиотека профессионала. Java 2. Том 1. Основы, 7-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс",

2007. – 896 с.

15. Хорстманн, К., С., Корнелл, Г. Библиотека профессионала. Java 2. Том 2. Тонкости программирования, 7-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2007. – 1168 с.

16. Бертран Мейер «Почувствуй класс. Учимся программировать хорошо с объектами и контрактами».

17. Гради Буч «Объектно-ориентированный анализ и проектирование с примерами приложений» .

18. "Head First Java, Изучаем Java", Кэти Сьерра, Берт Бэйтс.

19. Дж. Кьоу Объектно-ориентированное программирование / Дж. Кьоу, М. Джеанини. - М.: Питер, 2015. - 240 с.

20. Т. Пратт, М. Зелкович. Языки программирования. Разработка и реализация. 4-и издание. 688 с.

21. Роберт У. Себеста. Основные концепции языков программирования. 5-е издание. 668 с.

22. Гради Буч. Объектно-ориентированное программирование с примерами применения на C++. 2-е издание. 558 с.

23. Бентли Д. Жемчужины программирования, 2 изд. Питер, 2002 – 272 с.

24. Леффингуэл Д. Принципы работы с требованиями к программному обеспечению. Издательский дом «Вильямс», 2002 – 448 с.

25. Леффингуэл Д. Принципы работы с требованиями к программному обеспечению. Издательский дом «Вильямс», 2002 – 448 с.

26. https://vertex-academy.com/tutorials/uk/masivi-v-java/

ДОДАТКИ

Додаток А

Текст програми

```
package com.company;


public class MyString {
    public char[] myStr;
    public Integer length;
    public String newString;

    public MyString(String str) {

        this.myStr = str.toCharArray();
        this.length = this.myStr.length;
    }

    public char[] getMyStr() {
        return this.myStr;
    }

    public Integer length() {
        return this.myStr.length;
```

```
}

public String toUpperCase() {

    this.newString = new String();
    for (int i = 0; this.myStr.length > i; i++) {
        if (Character.isUpperCase(this.myStr[i])) {
            newString += this.myStr[i];
        }
        if (Character.isLowerCase(this.myStr[i])) {
            newString += Character.toUpperCase(this.myStr[i]);
        }
        if (this.myStr[i] == ' ') {

            newString += ' ';
        }

    }
    return newString;

}

public String toLowerCase() {

    this.newString = new String();

    for (int i = 0; this.myStr.length > i; i++) {
        if (Character.isLowerCase(this.myStr[i])) {
            newString += this.myStr[i];
        }
```

```
        if (Character.isUpperCase(this.myStr[i])) {
            newString += Character.toLowerCase(this.myStr[i]);
        }
        if (this.myStr[i] == ' ') {

            newString += ' ';
        }
    }
    return newString;
}

public char[] Concate(MyString string_2) {
    int len = this.myStr.length + string_2.myStr.length;
    char[] Concate = new char[len];
    int position = 0;
    for (char object : myStr) {
        Concate[position] = object;
        position++;
    }
    for (char object : string_2.myStr) {
        Concate[position] = object;
        position++;
    }
    this.myStr = Concate;
    return Concate;



}
public String substring (int start, int end) {
    if (start < 0 || end > this.myStr.length) {
```

```
            return "Bad parameters";
        }
        String string = new String(this.myStr);
        return string.substring(start, end);
    }

    public String substringEdit (int start, int end, String insertString) {
        String leftString = new String(this.myStr).substring(0, start);
        String rightString = new String(this.myStr).substring(end, this.myStr.length);
        return String.format("%s%s%s", leftString, insertString, rightString);
    }
}
```

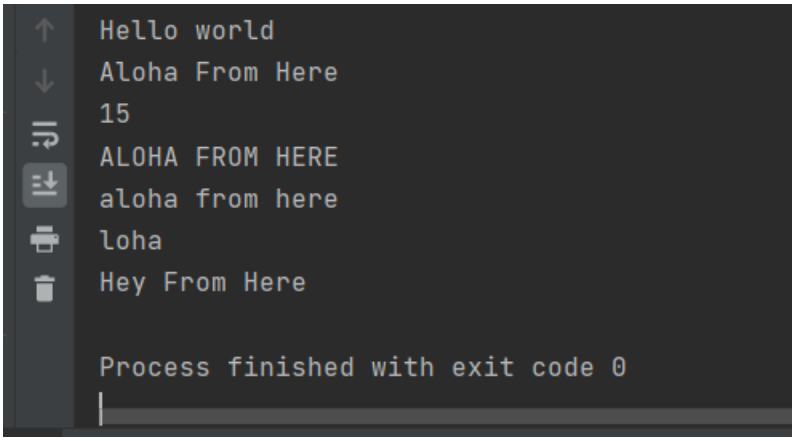
```
package com.company;

public class Main {
    public static void main(String[] args) {
        MyString str = new MyString("Aloha From Here");
        MyString gtr = new MyString("Hello world");

        System.out.println(gtr.getMyStr());
        System.out.println(str.getMyStr());
        System.out.println(str.length());
        System.out.println(str.toUpperCase());
        System.out.println(str.toLowerCase());
        System.out.println(str.substring(1, 6));
        System.out.println(str.substringEdit(0, 5, "Hey"));

    }}
```

Малюнок: результат виконання програми



ВИСНОВОК

В ході курсової роботи я на практиці застосував свої знання з об'єктно-орієнтованого програмування. Поглибив розуміння основних принципів об’єктно- орієнтованої ідеології програмування. Відпрацював на практиці методи та засоби ООП. Під час курсової роботи були реалізовані всі методи, які були у завданні мого варіанту.

В завданні потрібно було створити рядок в масиві, а для роботи з рядками в Java використовується клас String. Об'єкт класу String дозволяє виконувати різні операції над рядками символів. Об'єкт (змінну) типу String можна присвоювати іншому об'єкту типу String. Перед використанням, об'єкт (змінна) типу String обов'язково має бути ініціалізована.

Я закріпив свої знання та вміння з об’єктно-орієнтованого програмування, повторив як працювати з масивами, виклик елементів по індексу, реалізація всіх методів власноруч.

Також використав вміння програмувати мовою Java.