Лабораторна робота 1: Алгоритми сортування

Мета роботи:

1) дослідити порівняльні алгоритми сортування та алгоритми сортування за лінійний час
2) набути практичних навичок зі створення та обробки одно- та двовимірних масивів об'єктів на мові програмування Java
3) набути практичних навичок із порівняння вмісту об'єктів і реалізації сортування масивів об'єктів на мові програмування Java
4) набути практичних навичок із сортування зв'язаних списків на мові програмування Java
5) навчитися підключати і використовувати бібліотеки сторонніх розробників у власному проекті;
6) навчитися аналізувати результати сортування масивів і списків на різних наборах тестових даних.

Необхідні теоретичні знання мови Java

1. Опис полів і методів класу:
   https://docs.oracle.com/javase/tutorial/java/javaOO/classes.html
2. Робота з масивами об'єктів:
   http://www.javawithus.com/tutorial/array-of-objects
3. Форматування даних:
   https://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html
4. Перетворення рядка в числові типи:
   https://docs.oracle.com/javase/tutorial/java/data/numberclasses.html
5. Перевірка символів:
   http://docs.oracle.com/javase/7/docs/api/java/lang/Character.html

Інструментальні і програмні засоби:

Java JDK 8
IDE Eclipse

Склад матеріалів для виконання лабораторної роботи

Пакет lab1

Каркас програми для виконання першого завдання: **`SorterTask1.java`**

Клас, який описує дані про студента: **`Student.java`**

Файл з даними про студентів: **`data/students.cvs`**

Бібліотека для роботи з **`csv`** *файлами:* **`lib/opencsv-3.8.jar`**

Загальні рекомендації:

Індивідуальні завдання наведені у розділі «Варіанти індивідуальних завдань до лабораторної роботи 1».
Робота складається з двох завдань, пов'язаних із сортуванням послідовностей: масивів та/або зв'язаних списків. Каркас коду для першого завдання знаходиться у файлі **`SorterTask1.java`**. Для другого завдання каркас коду не надається.

В обох завданнях дані для сортування програмно представлені об'єктами класу **`Student`**. Цей клас описується у файлі **`Student.java`**. Склад полів цього класу визначається варіантом індивідуального завдання (табл. 1 у стовпчик 2). У каркасі коду за приклад взяті поля **`name`** і **`course`**.

Дані, які містяться в об'єктах класу **`Student`**, формують структуру даних, елементи якої в подальшому упорядковуються згідно з завданням до лабораторної роботи. У першому завданні такою структурою виступає одновимірний масив, а у другому – одновимірний масив, двовимірний масив або список.

Дані про студентів зчитуються із зовнішнього файлу. Перед початком роботи програми дані записуються у файл за допомогою будь-якого текстового редактора, додатка MS Excel або засобами IDE Eclipse. Кожен рядок файлу містить дані про одного студента. Дані у рядку розділяються комами. Такий файловий формат для представлення табличних називається **`csv`** (від англ. comma-separated values – 'значення, розділені комою') [1]. Наприклад, дані про студентів (прізвище та номер курсу) зберігаються у файлі так:

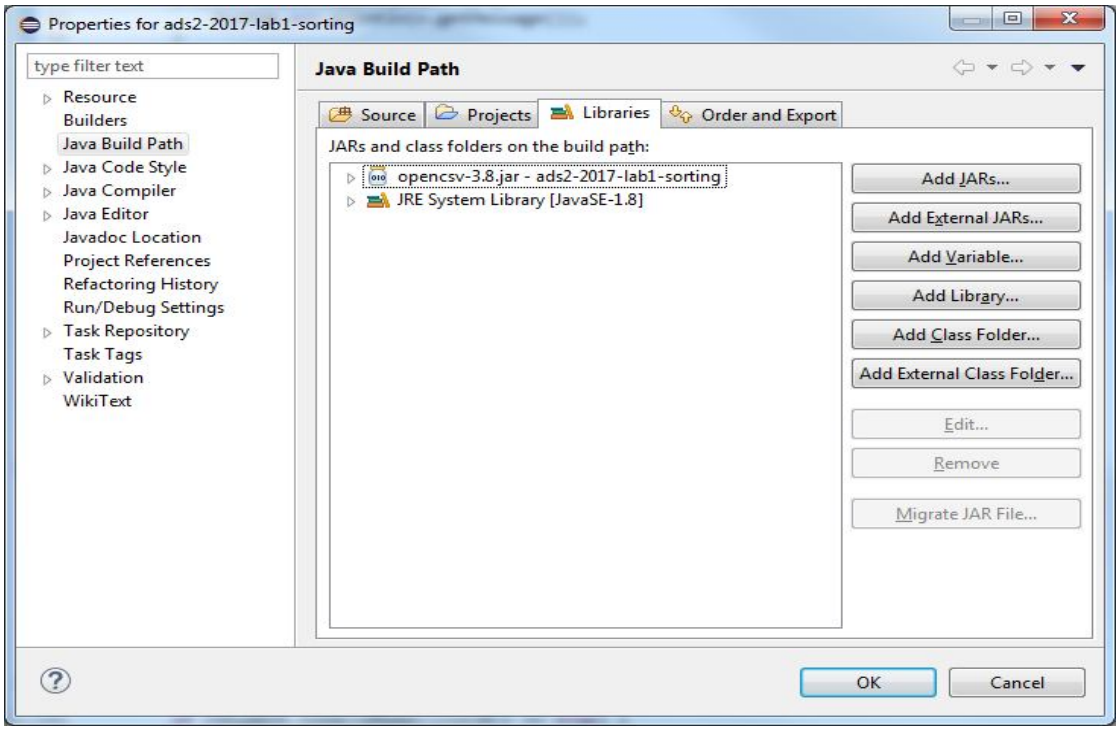
```
Tarasenko, 3
Melnik, 4
Kovalenko, 1
```

У складі проекту файл з даними про студентів називається **`"students.csv"`** і розташовується в директорії **`"data"`** поточного проекту. Перед виконанням роботи треба створити директорію **`"data"`**, створити файл **`"students.csv"`** і записати до нього дані про студентів. Назву файлу і його розташування можна змінити у цих рядках:

```
private static String fileName = "students.csv";
private static String currenDir = System.getProperty("user.dir")
                + File.separatorChar + "data";
```

Зверніть увагу, що порядок, в якому слідують дані, розділені комою, у кожному рядку файлу обов'язково має бути однаковим. Порядок розташування цих даних буде у подальшому враховуватися при зчитуванні їх з файлу, обробці та запису в поля об'єктів **`Student`**.

Для зчитування даних з файлу у форматі **csv** у програмі використовується бібліотека **opencsv-3.8.** Файл бібліотеки **opencsv-3.8.jar**, який треба приєднати до проекту, можна завантажити з сайту розробників вільного програмного забезпечення **SourceForge.net** [2] або з директорії **lib** матеріалів до лабораторної роботи.
У директорії поточного проекту треба створити директорію **lib** і зберегти в цій директорії завантажений файл **opencsv-3.8.jar.** Далі треба приєднати цей **JAR** файл до поточного проекту (з контекстного меню поточного проекту обрати **Build Path → Configure Build Path →** кнопка **Add JARs...**) [3]:



Оскільки у програмі використовується класи бібліотеки **opencsv,** то після оголошення назви пакету файл повинен містити рядок з імпортом пакету, де знаходяться імпортовані класи. Наприклад, імпорт класу **CSVReader** з пакету **com.opencsv**:

```
import com.opencsv.CSVReader;
```

В API-документації бібліотеки міститься докладний опис всіх класів і методів [4].

Метод main() в класі **SorterTask1** (файл **SorterTask1.java**) містить послідовність дій для виконання першого завдання лабораторної роботи: зчитування інформації з файлу, формування масиву об'єкті об'єктів класу **Student** і викликів методів для виконання завдання. Порядок дій у методі main() змінювати ***не треба.***

У першому завданні лабораторної сортується масив об'єктів класу **Student.** Дані, які сортуються зчитуються з файлу **"students.csv",** і формують елементи масиву. Якщо файл пустий, масив матиме нульову довжину, що в подальшому не призводить до виняткових ситуацій.

Всі рядки файлу зчитуються у список. Кожен елемент списку – це масив рядків. А кожен елемент масиву містить дані, які у раз їх коректності записуватимуться в поля об'єкту класу **`Student.`** Саме тому дуже важливо дотримуватися певного порядку розташування даних у файлі. Після того, як з файлу зчиталися всі рядки, в циклі аналізується кожен елемент списку (масив рядків): чи всі елементи масиву коректні з точки зору інформації про студента. Наприклад, ім'я студента повинно містити літери і починатися із великої літери, курс – ціле значення в діапазоні від 1 до 6. Методи для перевірки даних на коректність описані як статичні в класі **`Student`**. Для перевірки символів можна використовувати методи класу **`Character`**. Для перевірки не треба використовувати регулярні вирази.

Якщо всі дані про студента коректні, об'єкт класу **`Student`** стає елементом масиву, що буде сортуватися. Змінна **`numStudents`** використовується для підрахунку реальної кількості елементів, що сортуватимуться. Оскільки кількість **`numStudents`** може бути меншою, ніж кількість зчитаних рядків, то створюється копія масиву, яка містить лише коректні дані.

Масив перед сортуванням і після сортування треба вивести у формі таблиці, де рядок містить дані про одного студента. (метод **`printStudents`**) Перший стовпчик таблиці має містити ключ, за яким сортується масив. Наприклад, дані, зчитані з файлу **`"students.csv"`** у масив виводяться так:

```
--------------------
name      |  course|
--------------------
Tarasenko |  3     |
Melnik    |  4     |
Kovalenko |  1     |
--------------------
```

Файлі **`Student.java`** містить опис класу **`Student`**. Клас складається з двох полів (**`name`** і **`course),`** методів **`setXXX`** для відповідних полів і статичних методів перевірки коректності **`isValidXXX`**, які застосовуються перед тим, як встановити значення відповідного поля. Клас наведено як приклад, тому задля виконання першого і другого завдання його вміст описати, виходячи із даних у варіанті індивідуального завдання (табл.1, стовчик 2).

Клас **`SorterTask1`** (файл **`SorterTask1.java`**) містить методи, в яких є тег `// TODO`. Це означає, що тут ви маєте написати власний код. Сигнатуру (тип значення, що повертається, і параметри) існуючих методів змінювати ***не треба***. Але, звичайно, нові методи для реалізації програми можна дописати власноруч. Для нових методів потрібно створити документуючи коментарі з анотаціями @param @return. Щоб автоматично створити згенерувати заготовку коментарів для метода, необхідно стати на заголовок метода та обрати пункт меню **`Source → Generate Element Comment`**

Для виконання другого завдання не надається каркас коду. Програму треба написати самостійно. Клас **Student** не змінюється. Якщо сортується зв'язаний список, то вузол містить об'єкт класу **Student** і посилання на сусідні вузли.

Опис класів пакету lab1

Клас **Student** представляє опис інформації про студента і містить *поля*:

name – рядок, що зберігає ім'я студента;

course – ціле значення в діапазоні від 1 до 6;

методи:

public void setName(String name) – встановлює значення поля **name**

public void setCourse(int course) – встановлює значення поля **course**

статичні методи:

public static boolean isValidName(String name) – перевіряє, чи є рядок **name** коректним ім'ям: всі символи є літери, а перша літера - прописна

public static boolean isValidCourseNumber(String courseStr) – перевіряє, чи є рядок **courseStr** коректним номером курсу: у рядку один цифровий символ зі значенням у діапазоні від 1 до 6

public void print() – виводить дані про студента (значення полів) в одному рядку у форматованому вигляді

Опис класу наведено як приклад. Для виконання роботи початковий код класу треба переписати, виходячи з варіанту індивідуального завдання (табл.1, стовпчик 2).

Клас **SorterTask1** є головним класом програми для виконує першого завдання. Містить *статичні поля*:

fileName – назва файлу, звідки зчитуються дані

currenDir – назва директорії, де знаходиться файл **fileName**

Метод **main** становить порядок виконання програми. Цей порядок докладно описано у розділі «Загальні рекомендації».

Опис статичних методів класу SorterTask1

SLNode Student[] createArrayOfStudents(List<String[]> list) – створює і повертає масив об'єктів **Student,** перевіряючи коректність кожного елемента списку **list**. Для перевірки елемента списку і створення об'єкту класу **Student** викликається метод **writeStudInfo**. Об'єкти класу **Student** утворюють масив, що повертається з методу. Якщо кількість об'єктів з коректними даними **numStudents** менша, ніж кількість зчитаних рядків, то створюється копія масиву, яка містить лише коректні дані. Якщо жоден з елементів списку не можна використати для створення об'єктів класу **Student**, то метод повертає масив нульової довжини

`Student[] copyOf(Student[] students, int numStudents)` – створює новий масив об'єктів **`Student`** довжиною **`numStudents`** і копіює до нього **`numStudents`** перших елементів масиву **`students.`** Викликається у методі **`createArrayOfStudents.`** Метод містить тег **`TODO`**, тобто код методу треба написати самостійно.

`int writeStudInfo(Student s, String[] line)` – перевіряє всі рядки в масиві **`line`** на коректність і записує коректні дані у поля об'єкта **`s.`** Кожен з рядків відповідає певному полю класу **`Student`**. Для перевірки коректності цих рядків викликаються статичні методи **`isValidXXX`** класу **`Student.`** Коректний рядок записується у відповідне поле об'єкту класу. Якщо у всі поля об'єкта **`s`** записані коректні дані, то метод повертає 1, у протилежному випадку - 0. Метод містить тег **`TODO`**, тобто код методу треба написати самостійно.

`void printStudents(Student[] studs)` – виводить масив студентів у вигляді таблиці. Для виведення інформації про одного студента використовується метод **`print`** класу **`Student.`** Метод містить тег **`TODO`**, тобто код методу треба написати самостійно

`void sort(Student[] studs)` – сортує масив студентів за алгоритмом відповідно до варіанту індивідуального завдання (табл.1, стовпчик 3). Для порівняння об'єктів класу **`Student`** викликається метод **`compare`**. Метод містить тег **`TODO`**, тобто код методу треба написати самостійно

`int compare(Student s1, Student s2)` – порівнює об'єкти **`s1`** і **`s2`** відповідно до критерію сортування (табл.1, стовпчик 4). Повертає -1, якщо **`s1`** < **`s2`**, 1, якщо **`s1`** > **`s2`**, 0, якщо об'єкти рівні. Метод містить тег **`TODO`**, тобто код методу треба написати самостійно

`void swap(Student[] studs, int i, int j)` – міняє місцями два елементи масиву **`studs`** на позиціях **`i`** та **`j`**. Якщо номера позицій рівні, від'ємні або більше, ніж довжина масиву, масив не змінюється. Метод містить тег **`TODO`**, тобто код методу треба написати самостійно

Сигнатуру (параметри і тип значення, що повертається) наведених вище методів змінювати ***не треба***. За потреби в класі можна описати додаткові методи, які викликатимуться з тих, що описані.

Завдання

Описати клас **`Student`** з полями відповідно до варіанту завдання (табл.1, стовпчик 2). Для кожного поля описати метод **`setXXX`** і статичний метод перевірки коректності **`isValidXXX.`** Взяти за приклад опис класу у файлі **`Student.java.`**

***Завдання 1.1**: сортувати одновимірний масив елементів типу Student алгоритмом, визначеним варіантом завдання*

Порядок виконання (визначено в методі main класу SorterTask1)

1) Створити файл з даними про студентів
2) Зчитати дані з файлу
3) Створити одновимірний масив, елементи якого належать типу Student і заповнити його коректними даними (метод **`createArrayOfStudents`**)
4) Вивести вміст одновимірного масиву перед сортуванням (метод **`printStudents`**)
5) Упорядкувати масив за алгоритмом згідно з варіантом завдання (табл.1, кол.3) за заданим критерієм (табл.1, кол. 4) (метод **`sort`**)
6) Вивести вміст одновимірного масиву після сортування
7) Працездатність програми перевіряється на тестових наборах, різних за кількістю і характером упорядкування. Для тестових даних треба створити декілька файлів

***Завдання 1.2**: сортувати послідовність елементів типу Student алгоритмом, визначеним варіантом завдання*

Порядок виконання

1) Створити файл з даними про студентів
2) Зчитати дані з файлу
3) Створити послідовність *(*масив або зв'язаний список), елементи якої належать типу **`Student`** і заповнити її коректними даними
4) Вивести вміст послідовності перед сортуванням
5) Упорядкувати послідовність за алгоритмом згідно з варіантом завдання (табл.1, кол.3) за заданим критерієм (табл.1, кол. 4) (метод **`sort`**)
6) Вивести вміст послідовність після сортування
7) Працездатність програми перевіряється на тестових наборах, різних за кількістю і характером упорядкування. Для тестових даних треба створити декілька файлів

Посилання:

1. Стаття про файловий формат **CSV** у Вікіпедії: https://uk.wikipedia.org/wiki/CSV
2. Сторінка проекту на сайту розробників вільного програмного забезпечення **SourceForge.net:** https://sourceforge.net/projects/opencsv/files/opencsv/
3. Стаття «Як додати бібліотеки JAR проект на Eclipse (Java)»: http://help-me.pp.ua/25762-dodati-bbloteki-jar-proekt-na-eclipse-java.html
4. API-документація до **opencsv**: http://opencsv.sourceforge.net/apidocs/index.html
5. API-документація класу java.lang.Character: http://docs.oracle.com/javase/7/docs/api/java/lang/Character.html

Питання для самоперевірки:

1. Загальна характеристика елементарних алгоритмів сортування
2. Основний принцип, псевдокод і характеристика алгоритму сортування прямим обміном (бульбашки)
3. Способами можна зменшити час сортування прямим обміном
4. Основний принцип, псевдокод і характеристика алгоритму сортування вставкою
5. Основний принцип, псевдокод і характеристика алгоритму сортування вибіркою
6. Основний принцип, псевдокод і характеристика класичного алгоритму сортування Шела
7. Основний принцип, псевдокод і характеристика алгоритму сортування розподіленого підрахунку
8. Основний принцип, псевдокод і характеристика алгоритму порозрядного сортування за молодшими розрядами (LSD)
9. Основний принцип, псевдокод і характеристика алгоритму сортування комірками

Таблиця 1. Варіанти індивідуальних завдань лабораторної роботи 1 (частина 1)

Варіант	Дані про студента (поля класу «Студент»)	Алгоритм сортування	Критерій сортування
1	2	3	4
1.	Прізвище, ім'я, маса тіла, зріст	Шелла (класичний)	За зростанням маси тіла
2.	Прізвище, ім'я, група, середній бал, кількість пропущених занять за семестр	Бульбашки (прямим обміном)	За спаданням середнього балу та зростанням кількості пропущених занять
3.	Прізвище, ім'я, форма навчання (бюджетна/контрактна)	Вибіркою	За прізвищем в алфавітному порядку студентів кожної форми навчання
4.	Прізвище, ім'я, група, назва факультету	Вставкою	За назвою факультету в алфавітному порядку і за зростанням номера групи у кожному факультету
5.	Прізвище, ім'я, по-батькові, номер мобільного телефону	Шелла(за Кнутом)	За оператором мобільного зв'язку
6.	Прізвище, ім'я, номер студентського квитка, група	Бульбашки (прямим обміном)	За зростанням номеру групи і за прізвищем в алфавітному порядку у кожній групі
7.	Прізвище, ім'я, група, місце проживання – місто та область	Вибіркою	За назвою області і міста в алфавітному порядку
8.	Прізвище, ім'я, середній бал, стать	Двоспрямований бульбашковий	За зростанням середнього балу і за прізвищем в алфавітному порядку
9.	Прізвище, день, місяць і рік народження	Шелла (класичний)	За місяцем народження
10.	Прізвище, ім'я, середній бал, група	Бульбашки (прямим обміном)	За зростанням номера групи і спаданням середнього балу у кожній групі
11.	Прізвище, кількість всіх занять, кількість пропущених занять	Вставкою	За зростанням співвідношення пропущених і занять за планом
12.	Прізвище, ім'я, група, стать	Вибіркою	За зростанням номера групи та ім'ям в алфавітному порядку в кожній групі
13.	Прізвище, ім'я, група, номер студентського квитка	Шелла(за Кнутом)	За зростанням номера групи
14.	Прізвище, ім'я, група, номер залікової книжки	Вставкою	За зростанням номера групи і зростанням номера залікової книжки у кожній групі
15.	Прізвище, ім'я, код міста (тризначне число), номер домашнього телефону	Шелла(за Кнутом)	За кодом міста
16.	Прізвище, ім'я, номер кімнати у гуртожитку	Двоспрямований бульбашковий	За зростанням номера кімнати
17.	Прізвище, ім'я, дата народження	Шелла (класичний)	За зростанням віку
18.	Прізвище, ім'я, курс, назва ВНЗ, місто	Двоспрямований бульбашковий	За містом в алфавітному порядку і за назвою ВНЗ у кожному місті
19.	Прізвище, ім'я, факультатив, бал	Шелла(за Кну-	За зростанням балу навчання

Варіант	Дані про студента (поля класу «Студент»)	Алгоритм сортування	Критерій сортування
1	2	3	4
		том)	на факультативі
20.	Прізвище, ім'я, ідентифікаційний код, місце проживання	Двоспрямований бульбашковий	За спаданням ідентифікаційного коду
21.	Прізвище, ім'я, група, кількість заборгованостей	Бульбашки (прямим обміном)	За зростанням номера групи і за спаданням кількості заборгованостей у кожній групі
22.	Прізвище, ім'я, група, кількість балів при вступі у ВНЗ	Вибіркою	За зростанням номера групи і зростанням кількості балів у кожній групі
23.	Прізвище, ім'я, по-батькові, номер залікової книжки, ознака проходження військової підготовки (так/ні)	Вставкою	За зростанням номера залікової книжки серед тих, хто проходить не проходить військову підготовку
24.	Прізвище, ім'я, група, кількість виконаних лабораторних робіт	Двоспрямований бульбашковий	За спаданням кількості виконаних робіт

Таблиця 2. Варіанти індивідуальних завдань лабораторної роботи 1 (частина 2)

Варіант	Структура даних	Алгоритм сортування	Порядок сортування
1	2	3	4
1.	Одновимірний масив	Комірками із сортуванням кожного карману	За зростанням маси тіла
2.	Односпрямований список	За посиланням (з використанням додаткового масиву)	За спаданням середнього балу та зростанням кількості пропущених занять
3.	Двоспрямований список	Вставкою	За прізвищем в алфавітному порядку студентів кожної форми навчання
4.	Односпрямований список	Вибіркою	За назвою факультету в алфавітному порядку і за зростанням номера групи у кожному факультету
5.	Одновимірний масив	Порозрядний за молодшими розрядами	За оператором мобільного зв'язку
6.	Двовимірний масив (рядок – група)	Вставкою (для сортування елементів рядка)	За зростанням номеру групи і за прізвищем в алфавітному порядку у кожній групі
7.	Односпрямований список	Вибірки	За назвою області і міста в алфавітному порядку
8.	Двоспрямований список	За посиланням (з використанням додаткового масиву)	За зростанням середнього балу і за прізвищем в алфавітному порядку
9.	Одновимірний масив	Розподіленого підрахунку	За місяцем народження
10.	Одновимірний масив	Комірками зі вставкою елемента у сортований список у комірці	За зростанням номера групи і спаданням середнього балу у кожній групі
11.	Одновимірний масив	Комірками	За зростанням співвідношення

Варіант	Структура даних	Алгоритм сортування	Порядок сортування
1	2	3	4
			пропущених і занять за планом
12.	Двовимірний масив (рядок – група)	Шелла (для сортування елементів рядка)	За зростанням номера групи та ім'ям в алфавітному порядку в кожній групі
13.	Одновимірний масив	Порозрядний за молодшими розрядами	За зростанням номера групи (формат номеру групи ЛЛ-ЦЦ, де Л – літера, Ц – цифра)
14.	Двовимірний масив (рядок – група)	Шелла за Кнутом (для сортування елементів рядка)	За зростанням номера залікової книжки у кожній групі
15.	Одновимірний масив	Порозрядний	За кодом міста
16.	Одновимірний масив	Карманний (карман для кожної кімнати)	За зростанням номера кімнати
17.	Одновимірний масив	Розподіленого підрахунку	За зростанням віку
18.	Односпрямований список	Вибірки	За містом в алфавітному порядку і за назвою ВНЗ у кожному місті
19.	Двоспрямований список	За посиланням (з використанням додаткового масиву)	За зростанням балу навчання на факультативі
20.	Одновимірний масив	Порозрядний за молодшими розрядами (із сортування комірками)	За спаданням ідентифікаційного коду
21.	Двовимірний масив (рядок – група)	Розподіленого підрахунку (для сортування елементів рядка)	За зростанням номера групи і за спаданням кількості заборгованостей у кожній групі
22.	Двоспрямований список	Комірками (номер комірки – перша цифра в номері групи)	За зростанням номера групи і зростанням кількості балів у кожній групі (формат номеру групи КП-ЦЦ, де Ц – цифра)
23.	Односпрямований список	Вибірки	За зростанням номера залікової книжки серед тих, хто проходить і не проходить військову підготовку
24.	Одновимірний масив	Розподіленого підрахунку (із сортування комірками)	За спаданням кількості виконаних робіт