Лабораторна робота №4

Ввід-вивід. Контейнерні класи.

_____

Скачати з DL архів із заглушкою проекту. Архів розпакувати та імпортувати вміст у IDE (Eclipse / IDEA) як Maven project. Відповідно до завдання внести зміни до проекту. Досягнути проходження тестів.
Завантажити рішення на DL.

Зауваження.

1. Кореневий пакет: ua.nure.jfn.task4

2. Якщо програма зчитує інформацію з файлу, необхідно вказати кодування, в якому записана інформація. Використовувати кодування Unicode у форматі UTF-8. Розділювач рядків у файлах \r\n або \n.

3. Кожен клас PartX повинен мати метод main, який демонструє роботу відповідного підзавдання.У кореневому пакеті створити клас Demo, який демонструє роботу всього написаного функціоналу.

4. Для того щоб продемонструвати функціональність, яка потребує консольного введення, необхідно моделювати стандартний потік введення (STDIN). Перед викликом демонстраційного коду слід замінити стандартний потік введення на вхідний потік, який містить демонстраційні дані. Після того як відпрацює функціональність, необхідно відновити стандартний потік введення.

Метод Demo.main має виконуватись повністю автоматично, без участі користувача – очікування введення з консолі під час його виконання не допускається.

5. Не закривайте потоки, надбудовані над системними (standard input/output), т.к. це спричинить закриття системних потоків і зробить їх непридатними для подальшого використання. Клас Demo повинен демонструвати роботу всіх підзадач і якщо якась із них закриє системний потік, то наступні частини не зможуть з ним працювати.

6. Всі роботи будуть перевірені на плагіат. Якщо буде детектовано плагіат, оцінку буде знижено.

7. Робота буде оцінена в діапазоні [0..100*k], де k залежить від форми здачі л.р.
Варіанти:

(1) Залити архів із рішенням на DL: k=0.7
Максимальна оцінка – 70 балів.

(2) Залити архів із рішенням на DL, а також залити текстовий файл з назвою link.txt (кодування файлу Unicode у форматі UTF-8), в якому розмістити посилання на відеозапис з поясненням, як було вирішено завдання: k=0.9
Максимальна оцінка – 90 балів.

(3) залити архів із рішенням на DL, захистити л.р. шляхом спілкування з викладачем: k=1
Максимальна оцінка – 100 балів.

_____

Завдання 1

Назва класу: ua.nure.jfn.task4.Part1
Вхідну інформацію завантажувати із файлу part1.txt

Файл містить слова, записані кирилицею та латиницею, у кодуванні Unicode (формат UTF-8).
Необхідно написати клас, який має таку функціональність:
1) Якщо користувач вводить у консолі Latn, додаток виводить усі слова, записані латиницею.
2) Якщо вводиться Cyrl, додаток виводить усі слова, записані кирилицею.
3) Команда Stop або stop завершує роботу.
4) Будь-яке інше введення спричиняє появу повідомлення: Incorrect input.

Слово — це неперервна послідовність символів кирилиці або латиниці.

Вказану вище функціональність повинен містити метод Part1.main
Клас Demo, який у методі main демонструє роботу всіх частин, має моделювати консольне введення перед викликом Part1.main

Приклад вхідного файлу:

Is there anybody going to listen to my story
Лягає день. Він віддає свої надії ночі.
Робітники
Заморились працювати.

Приклад консолі:
---
asdf<Enter>
asdf: Incorrect input
Latn<Enter>
Latn: Is there anybody going to listen to my story
Cyrl<Enter>
Cyrl: Лягає день Він віддає свої надії ночі Робітники Заморились працювати
Stop

Завдання 2

Назва пакету: ua.nure.jfn.task4
Класи:
Part2 - містить моделювання консольного введення та виклик методу WordContainer.main
WordContainer - містить необхідну функціональність
Word - поставлений у відповідність до слова з інформацією про його частоту в тексті

Розробити клас WordContainer, який зчитує текст із консольного введення та виводить слова в порядку зростання частоти їх появи в тексті (у разі збігу частот — у лексикографічному порядку).

Під словом слід розуміти неперервну послідовність непробільних символів.

Розв’язати задачу з використанням об’єктно-орієнтованого підходу. Клас Word має містити рядкове поле content і ціле поле frequency. Клас WordContainer агрегує об’єкти Word.

Під час використання контейнерних класів з бібліотеки ядра реалізувати методи equals, hashCode, compareTo класу Word (за потреби).

Приклад вхідної інформації (консоль):
---
asd asdf asd asdf<Enter>
asdf 43 asdsf 43 43 434<Enter>
stop<Enter>

Результат:
---
434 : 1
asdsf : 1
asd : 2
43 : 3
asdf : 3

Під час запуску метод WordContainer.main зчитує дані зі стандартного потоку введення (консолі).

Консольне введення зазвичай може містити кілька рядків, у кожному з яких — кілька слів.

Ознакою завершення введення є слово stop.

Завдання 3

Назва класу: ua.nure.jfn.task4.Part3

Написати клас, який у циклі зчитує з консолі через пробіл ключ (key) та назву локалі; у відповідь виводить відповідне значення за ключем. Якщо значення за вказаним ключем знайти неможливо, програма повинна виводити рядок Incorrect input. Ознакою завершення введення є слово stop.

Базова назва пакетів ресурсів – resources. Усі пакети ресурсів є статичними .properties файлами, що знаходяться в корені classpath.
Для розв’язання задачі необхідно використовувати java.util.ResourceBundle та java.util.Locale.

Читання з консолі та вивід у консоль є обов’язковими.

Вказану вище функціональність повинен містити метод Part3.main
Клас Demo, який у методі main демонструє роботу всіх частин, має моделювати консольне введення перед викликом Part3.main

Наведено приклад для двох локалей (загалом їх може бути скільки завгодно).
У каталозі src/main/resources розміщено пакети ресурсів для локалей en і uk:

resources_en.properties

```
table = table
apple = apple
```

---

resources_uk.properties

```
table = \u0441\u0442\u0456\u043b
apple = \u044f\u0431\u043b\u0443\u043a\u043e
```

Результат роботи програми:

```
table en<Enter>
table
table uk<Enter>
стіл
apple en<Enter>
apple
asfd asdf<Enter>
Incorrect input
apple uk<Enter>
яблуко
stop
```

Зауваження. Кодування properties файлів повинно бути ISO-8859-1, UTF-8 не використовувати.

Завдання 4

Назва класу: ua.nure.jfn.task4.Part4
Вхідну інформацію завантажувати з файлу part4.txt
Вихідну інформацію завантажувати до файлу part4_sorted.txt

Створити клас, який створює і заповнює файл part4.txt (якщо він не існує) випадковими цілими числами від 0 до 50 (загалом 10 чисел) через пробіл, потім читає файл і виводить його вміст в

інший файл (part4_sorted.txt), відсортувавши числа за зростанням і видаливши дублікати.
Вміст обох файлів (числа розділені пробілами) вивести в консоль.

Зауваження. Вхідний файл part4.txt створює тільки при першому запуску, всі наступні виклики використовують вже існуючий файл.

Для сортування написати власний метод, який здійснює сортування деяким алгоритмом (наприклад "бульбашкою"). Файли part4.txt и part4_sorted.txt є текстовими, кодування Unicode (UTF-8).

Вивести вміст вхідного та вихідного файлу в консоль дотримуючись заданого формату:

```
-----------------------------------------------------
input  ==> 41 40 35 2 2 3 11 40 11 10
output ==> 2 3 10 11 35 40 41
-----------------------------------------------------
```

Завдання 5

```
-----------------------------------------------------
Назва пакету: ua.nure.jfn.task4
Назви класів: Part5, Tree
-----------------------------------------------------
```

Створити generic-клас Tree, який реалізує структуру даних, що стає бінарним деревом пошуку при інверсії відношення порядку.
Відношення порядку є строгим: дублікати не допускаються.
(див. визначення бінарного дерева пошуку у Wikipedia)

Контейнерні класи не використовувати.

Елементи класу Tree, обов'язкові для реалізації:

```
-----------------------------------------------------
public class Tree<E extends Comparable<E>> {

    // додає елемент у контейнер
    // якщо в контейнері є елемент рівний по compareTo,
    // то додавання не відбувається і метод повертає false
    // інакше елемент потрапляє у контейнер і метод повертає true
    // перший елемент, що додається, стає коренем дерева
    // автобалансування у дереві немає
    public boolean add(E element) {...}

    // додає всі елементи з масиву в контейнер (виклик у циклі методу add, див. вище)
    public void add(E[] elements) {...}

    // видаляє елемент із контейнера
    // якщо елемента, що видаляється, в контейнері немає, то повертає false
    // інакше видаляє елемент і повертає true
    // ВАЖЛИВО! при видаленні елемента дерево не повинно втратити властивості бінарного дерева пошуку
    public boolean remove(E element) {...}

    // роздруковує дерево, щоб було видно його деревоподібну структуру, див. нижче приклад
    // використовувати символи псевдографіки
    public void print() {...}

    // Вкладений клас, об'єкти цього класу становлять дерево
    private static class Node<E> {...}

}
-----------------------------------------------------
```

Продемонструвати роботу програми (Part5.main).

Приклад вмісту методу Part5.main:

```
-----------------------------------------------------
Tree<Integer> tree = new Tree<>();
```

```
System.out.println(tree.add(3));
System.out.println(tree.add(3));

System.out.println("~~~~~~~");
tree.add(new Integer[] { 1, 2, 6, 4, 7, 0, 5 });
tree.print();

System.out.println("~~~~~~~");
System.out.println(tree.remove(3));
System.out.println(tree.remove(3));

System.out.println("~~~~~~~");
tree.print();
```

результат, який генерує Part5.main:

```
true
false
~~~~~~~
    ┌0
  ┌1┤
  │ └2
3─┤
  │ ┌4┐
  │ │ └5
  └6┤
    └7
~~~~~~~
true
false
~~~~~~~
    ┌0
  ┌1┤
  │ └2
4─┤
  │ ┌5
  └6┤
    └7
```

Під час видалення вузла з дерева слід дотримуватися такого алгоритму:
1. Якщо вузол не має дочірніх вузлов, достатньо встановити у батьківському вузлі відповідне посилання на null.
2. Якщо вузол має лише один дочірній вузол, потрібно переадресувати посилання від батьківського вузла на цього дочірнього вузла, тим самим минаючи вузол, що видаляється.
3. Якщо вузол має два дочірні вузли, слід виконати наступні кроки:
3.1. знайти наступника (найменший елемент у лівому піддереві, цей вузол не має правого дочірнього вузла);
3.2. перемістити лівого дочірнього вузла наступника на місце самого наступника;
3.3. замінити вузол, що видаляється, знайденим наступником.

_____

Запитання.
1. Метод equals: де визначено, призначення, контракт (5 пунктів).
2. Колекції: ієрархія інтерфейсів і класів.
3. Списки (List): які є, у чому відмінності?
4. Типи Iterator/Iterable: у яких пакетах визначені, що містять.
5. Множини (Set): які є, у чому відмінності?
6. Comparable/Comparator: чи параметризовані, що містять, контракт методів.
7. Метод hashCode: де визначено, контракт, зв’язок з equals.
8. Зв’язок між equals і compareTo/compare.
9. Карти (Map): інтерфейси, класи, у чому відмінності?
10. Collection / Collections: різниця.