Лабораторна робота №2

ООП, класи, інтерфейси.

_____

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.task2

2. Додатково створити у кореневому пакеті клас Demo, який демонструє роботу всього створеного функціоналу.

3. Ви повинні знати відповіді на запитання, які наведені наприкінці тексту (це допуск до л.р.).

4. Після того, як завдання буде зроблено, необхідно домогтися проходження тестів; чим більше тестів буде у статусі "passed", тим вище буде підсумкова оцінка.

5. Сформувати архів проекта за образом та подобою вхідного проекту заглушки, та завантажити на DL (кореневий каталог архіву містить лише каталог src та файл pom.xml). Архів не повинен містити бінарні файли (*.class).

6. Заборонено використовувати типи з пакета java.util та його підпакетів крім:
java.util.Arrays
java.util.Iterator
java.util.NoSuchElementException

Також заборонено використовувати контейнерні класи ядра (списки, множини, асоціативні контейнери і т.п.) з інших пакетів.

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

8. 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.

_____

Створити та розташувати в кореневому пакеті (ua.nure.jfn.task2) інтерфейс Container:

```
------------------------------
package ua.nure.jfn.task2;

import java.util.Iterator;

public interface Container<T> extends Iterable<T> {

    // Appends the specified element to the end.
    void add(T element);

    // Removes all of the elements.
    void clear();

    // Returns the number of elements.
```

```
    int size();

    // Returns a string representation of this container.
    // See JUnit tests for details.
    String toString();

    // Returns an iterator over elements.
    // Iterator must implement the remove method.
    Iterator<T> iterator();

    // Returns a stream over elements.
    Stream<T> stream();

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

Створити та розташувати в кореневому пакеті (ua.nure.jfn.task2) інтерфейс Stream:

```
-----------------------------
package ua.nure.jfn.task2;

// A sequence of elements.
public interface Stream<T> {

    // Represents a function that accepts one argument and produces a result.
    // X - the type of the input to the function;
    // Y - the type of the result of the function.
    interface Function<X, Y> {
        Y apply(X x);
    }

    // Represents an operation that accepts a single input argument
    // and returns no result.
    // X - the type of the input to the operation.
    interface Action<X> {
        void perform(X x);
    }

    // Intermediate operations

    // Returns a stream consisting of the elements of this stream that match
    // the given predicate.
    Stream<T> filter(Function<? super T, Boolean> predicate);

    // Returns a stream consisting of the results of applying the given function
    // to the elements of this stream.
    <R> Stream<R> map(Function<? super T, ? extends R> function);

    // terminal operations

    // Returns the count of elements in this stream.
    int count();

    // Performs an action for each element of this stream.
    void forEach(Action<? super T> action);

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

_____

Завдання 1 (Динамічний масив)

Ім'я пакету: ua.nure.jfn.task2
Імена типів: Array, ArrayImpl

1.1. Створити інтерфейс Array наступного змісту:

```
-----------------------------
package ua.nure.jfn.task2;
```

```
public interface Array<T> extends Container<T> {

    // Returns the element at the specified position.
    T get(int index);

    // Returns the index of the first occurrence of the specified element,
    // or -1 if this array does not contain the element.
    // (use 'equals' method to check an occurrence)
    int indexOf(T element);

    // Removes the element at the specified position.
    void remove(int index);

    // Sets the element at the specified position.
    void set(int index, T element);

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

1.2. Створити клас ArrayImpl, що реалізує інтерфейс Array.

Зберігання об'єктів контейнера реалізувати за допомогою масиву об'єктів.

Метод iterator повинен повертати екземпляр класу IteratorImpl, який реалізує інтерфейс java.util.Iterator<E>.
Клас IteratorImpl має бути визначений усередині класу ArrayImpl (є внутрішнім класом).

Якщо у контейнер були додані за допомогою методу add три елементи A, B, C, то:
1) метод toString повинен повертати рядок "[A, B, C]"
2) порядок обходу елементів контейнера ітератором: A B C

Метод stream повинен повертати екземпляр класу StreamImpl.

1.3. У методі main класу ArrayImpl продемонструвати роботу всіх методів інтерфейсу Array.

Завдання 2 (Зв'язаний список)

```
-----------------------------------------------------
Ім'я пакету: ua.nure.jfn.task2
Імена типів: List, ListImpl
-----------------------------------------------------
```

2.1. Створити інтерфейс List наступного змісту:

```
-----------------------------
package ua.nure.jfn.task2;

public interface List<T> extends Container<T> {

    // Inserts the specified element at the beginning.
    void addFirst(T element);

    // Removes the first element.
    void removeFirst();

    // Removes the last element.
    void removeLast();

    // Returns the first element.
    T getFirst();

    // Returns the last element.
    T getLast();

    // Returns the first occurrence of the specified element.
    // Returns null if no such element.
    // (use 'equals' method to check an occurrence)
    T search(T element);

    // Removes the first occurrence of the specified element.
```

```
    // Returns true if this list contained the specified element.
    // (use 'equals' method to check an occurrence)
    boolean remove(T element);

}
```

2.2. Створити клас ListImpl, який реалізує інтерфейс List.

Зберігання об'єктів контейнера реалізувати за допомогою сукупності вузлів – екземплярів класу Node.
Кожен вузол зберігає об'єкт та посилання на наступний вузол.
Клас Node повинен бути визначений усередині класу ListImpl (є вкладеним static класом).

Метод iterator повинен повертати екземпляр класу IteratorImpl, який реалізує інтерфейс java.util.Iterator<E>.
Клас IteratorImpl має бути визначений усередині класу ListImpl (є внутрішнім класом).

Якщо до контейнера були додані за допомогою методу addLast три елементи A, B, C, то:
1) метод toString повинен повертати рядок "[A, B, C]"
2) порядок обходу елементів контейнера ітератором: A B C

Метод stream повинен повертати екземпляр класу StreamImpl.

2.3. У методі main класу ListImpl продемонструвати роботу всіх методів інтерфейсу List.

Завдання 3 (Стрим)

Ім'я пакету: ua.nure.jfn.task2
Імена типів: Stream, StreamImpl

Створити клас StreamImpl, який реалізує інтерфейс Stream.

Запитання.

1. Що таке перекриття, перевантаження, приховування методу.
2. Що таке успадкування, ключові слова implements, extends.
3. Які типи не можуть бути успадковані?
4. Обмеження при перекритті методу.
5. Що таке інкапсуляція, для чого вона призначена, як її реалізувати.
6. Контексти використання ключового слова final.
7. Порядок виклику конструкторів, блоків ініціалізації при створенні об'єкта.
8. Анонімні класи.
9. В чому відмінність вкладених класів від внутрішніх.
10. Які типи не можуть бути узагальнені, написати приклад generic методу.