Лабораторна робота №2.

Мета

● Набуття навичок розробки власних контейнерів.

● Використання ітераторів.

● Тривале зберігання та відновлення стану об’єктів.

● Ознайомлення з принципами серіалізації/десеріалізації об’єктів.

● Використання бібліотек класів користувача.

Вимоги

1. Розробити клас-контейнер, що ітерується для збереження початкових даних Вашого варіанту завдання з попередньої роботи (Прикладні задачі. Список з 1-15 варіантів) у вигляді масиву рядків з можливістю додавання, видалення і зміни елементів.

2. В контейнері реалізувати та продемонструвати наступні методи: ○ String toString() повертає вміст контейнера у вигляді рядка; ○ void add(String string) додає вказаний елемент до кінця контейнеру; ○ void clear() видаляє всі елементи з контейнеру;

○ boolean remove(String string) видаляє перший випадок вказаного елемента з контейнера;

○ Object[] toArray() повертає масив, що містить всі елементи у контейнері;

○ int size() повертає кількість елементів у контейнері; ○ boolean contains(String string) повертає true , якщо контейнер містить вказаний елемент;

○ boolean containsAll(Container container) повертає true , якщо контейнер містить всі елементи з зазначеного у параметрах;

○ public Iterator<String> iterator() повертає ітератор відповідно до Interface Iterable .

3. В класі ітератора відповідно до Interface Iterator реалізувати методи:

○ public boolean hasNext() ;

○ public String next() ;

○ public void remove() .

4. Продемонструвати роботу ітератора за допомогою циклів while

и for each.

5. Забороняється використання контейнерів (колекцій) і алгоритмів з Java Collections Framework .

6. Реалізувати і продемонструвати тривале зберігання/відновлення розробленого контейнера за допомогою серіалізації/десеріалізації .

7. Обмінятися відкомпільованим (без початкового коду) службовим класом (Utility Class) рішення одного варіанту задачі (Прикладні задачі. Список з 1-15 варіантів) з сусіднім номером. 1 міняється з 2, 2 з 3, 3 з 4, 4 з 5 і т.д. Останній, 15 міняється з 1 варіантом і далі аналогічно.

8. Продемонструвати послідовну та вибіркову обробку елементів розробленого контейнера за допомогою власного і отриманого за обміном службового класу.

9. Реалізувати та продемонструвати порівняння, сортування та пошук елементів у контейнері.

10.Розробити консольну програму та забезпечити діалоговий режим роботи з користувачем для демонстрації та тестування рішення.

Розробник: Гринишин Анастасія , КН-108, номер варіанту

індивідуального завдання – 10.

Задача: Розробка власних контейнерів. Ітератори. Серіалізація/десеріалізація об’єктів. Бібліотека класів користувача .

Ієрархія та структура класів :

1.Клас Main, який містить 1 функцію – main

2.Клас Main_1,який містить 1 функцію - main_1(клас який звязує цей проект з попереднім)

3.Клас Audit, який містить 3 функції - how_much, make_table, ture.

3.Клас MyContainer, який містить всі вище перераховані функції з вимог і внутрішній клас Itr.

4.Клас Itr, який містить 3 функції – hasNext, next, remove.

5 Клас Param, який містить 2 функції – menu, help.

Важливий фрагмент

```
import java.io.Serializable;
import java.util.ConcurrentModificationException;
import java.util.Iterator;
import java.util.NoSuchElementException;

public class MyContainer<E> implements Iterable, Serializable {
    private final int INT_VAl =16;
    public String[] MAS = new String[INT_VAl];
    public int size =0;
    public int cursor;


    @Override
    public Iterator<String> iterator() {
        return new Itr();
    }

    private class Itr implements Iterator<String>{
        int cursor;       // index of next element to return
        int lastRet = -1; // index of last element returned; -1 if no such

        public boolean hasNext() { return cursor != size; }

        /unchecked/
        public String next() {
//            checkForComodification();
            int i = cursor;
            if (i >= size)
                throw new NoSuchElementException();
            String [] array = MyContainer.this.MAS;
            if (i >= array.length)
                throw new ConcurrentModificationException();
            cursor = i + 1;
            return (String) array[lastRet = i];
        }

        public void remove() {
            if (lastRet < 0)
                throw new IllegalStateException();
//            checkForComodification();

            try {
                MyContainer.this.remove(lastRet);
                cursor = lastRet;
                lastRet = -1;
            } catch (IndexOutOfBoundsException ex) {
                throw new ConcurrentModificationException();
            }
        }

        /*final void checkForComodification() {
            if (modCount != expectedModCount)
                throw new ConcurrentModificationException();
        }*/
    }



    // Метод для повернення елемента за індексом
    public String set(int index){
MyContainer › Itr
```

Висновок

У ході роботи дізналася про контейнери, як створювати їх і як застосовувати.