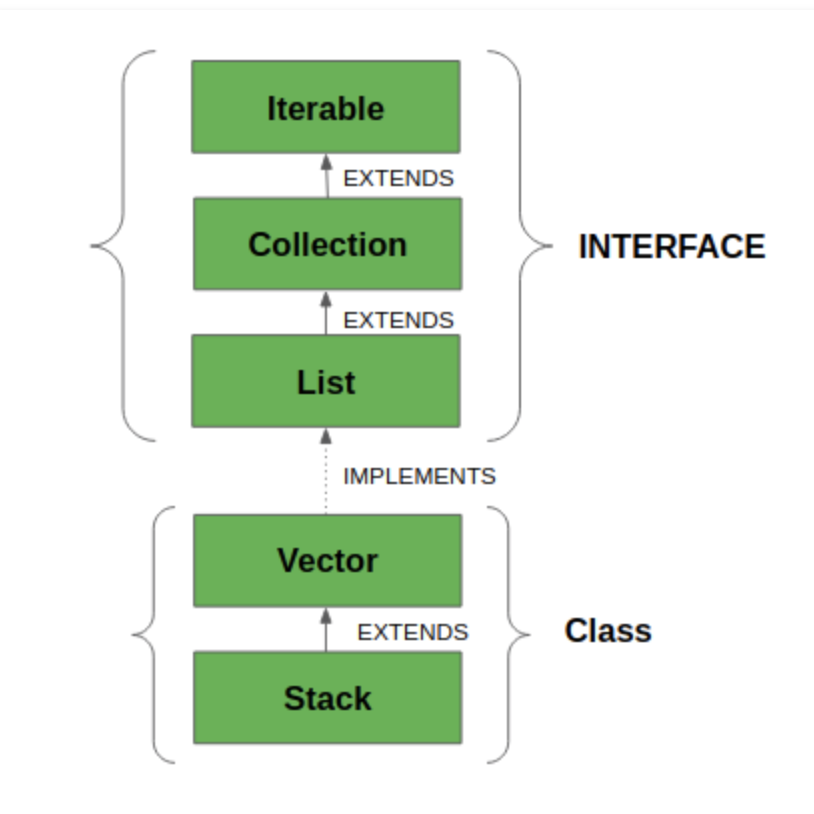
Практическая работа № 10

**ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ:**

Тема: Java Collections Framework

В состав Java Collection входит класс Stack, который моделирует и реализует структуру данных Stack. Класс основан на базовом принципе «первым пришел - первым вышел». В дополнение к основным операциям push и pop, класс предоставляет еще три функции empty, search и peek. Можно также сказать, что класс расширяет Vector и рассматривает класс как стек с пятью упомянутыми функциями. Этот класс также может называться подклассом Vector.

Эта диаграмма показывает иерархию класса Stack:



Итераторы используются Java Collections Framework

для последовательного доступа к элементам контейнера, с которым работает итератор. Для более подробной информации и введения, связанного с этим. Ниже представлен код Java программы, который демонстрирует работу с итератором.

import java.util.ArrayList;

import java.util.Iterator;

public class Test

{

    public static void main(String[] args)

    {

        ArrayList al = new ArrayList();

        for (int i = 0; i < 10; i++)

            al.add(i);

        System.out.println(al);

// at beginning itr(cursor) will point to

        // index just before the first element in al

        Iterator itr = al.iterator();

        // checking the next element availabilty

        while (itr.hasNext())

        {

            //  moving cursor to next element

            int i = (Integer)itr.next();

            // getting even elements one by one

            System.out.print(i + " ");

            // Removing odd elements

            if (i % 2 != 0)

               itr.remove();

        }

        System.out.println();

        System.out.println(al);

    }

}

Результат выполнения программы:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

0 1 2 3 4 5 6 7 8 9

[0, 2, 4, 6, 8]

Зачем нужно реализовывать интерфейс Iterable?

Каждый класс, который соответствующим образом реализует интерфейс Iterable, может использоваться в расширенном цикле For (для цикла for-each). Необходимость реализации интерфейса Iterator возникает при проектировании пользовательских структур данных.

Пример:

Программный код ниже иллюстрирует реализацию интерфейса Iterable

class CustomDataStructure implements Iterable<> {

    // code for data structure

    public Iterator<> iterator() {

        return new CustomIterator<>(this);

    }

}

Программный код ниже иллюстрирует реализацию интерфейса Iterator

class CustomIterator<> implements Iterator<> {

    // constructor

    CustomIterator<>(CustomDataStructure obj) {

        // initialize cursor

    }

    // Checks if the next element exists

    public boolean hasNext() {

    }

    // moves the cursor/iterator to next element

    public T next() {

    }

    // Used to remove an element. Implement only if needed

    public void remove() {

        // Default throws UnsupportedOperationException.

    }

}

Задания:

Упражнение 1

Реализуйте метод, следующим образом: исходный массив необходимо инвертировать, последовательно меняя местами 1 и последний элемент, 2 и предпоследний и так далее, для работы необходимо использовать контейнер Stack.

Упражнение 2

Написать свой собственный итератор для интерфейса List и создать его реализацию. Итератор (от англ. iterator ― перечислитель) — интерфейс, предоставляющий доступ к элементам коллекции (массива или контейнера) и навигацию по ним.

Упражнение 3

Напишите реализацию Iterator для вашего собственного List