



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения

**ОТЧЁТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №7
по дисциплине
«Программирование на языке Java»**

**Тема: Использование стандартных контейнерных классов при
программировании на Java**

Выполнил студент группы ИКБО-16-20

Пак С.А.

Принял ассистент кафедры ИиППО

Русляков А.А.

Практические работы выполнены «_____» 2021г.

«Зачтено» «_____» 2021г

Москва 2021

СОДЕРЖАНИЕ

ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ.....	3
РЕШЕНИЕ ЗАДАЧИ.....	4
1. Постановка задачи.....	4
2. Программный код.....	4
3. Вывод программы.....	10
ВЫВОД.....	11

ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

Java Collections Framework — это набор связанных классов и интерфейсов, реализующих широко используемые структуры данных — коллекции. На вершине иерархии в Java Collection Framework располагаются 2 интерфейса: Collection и Map. Эти интерфейсы разделяют все коллекции, входящие в фреймворк на две части по типу хранения данных: простые последовательные наборы элементов и наборы пар «ключ — значение» (словари).

Vector — реализация динамического массива объектов. Позволяет хранить любые данные, включая null в качестве элемента. Vector появился в JDK версии Java 1.0, но, как и Hashtable, эту коллекцию не рекомендуется использовать, если не требуется достижения потокобезопасности. Потому как в Vector, в отличие от других реализаций List, все операции с данными являются синхронизированными. В качестве альтернативы часто применяется аналог — ArrayList.

Stack — данная коллекция является расширением коллекции Vector. Была добавлена в Java 1.0 как реализация стека LIFO (last-in-first-out). Является частично синхронизированной коллекцией (кроме метода добавления push()). После добавления в Java 1.6 интерфейса Deque, рекомендуется использовать именно реализации этого интерфейса, например ArrayDeque.

ArrayList — как и Vector является реализацией динамического массива объектов. Позволяет хранить любые данные, включая null в качестве элемента. Как можно догадаться из названия, его реализация основана на обычном массиве. Данную реализацию следует применять, если в процессе работы с коллекцией предполагается частое обращение к элементам по индексу. Из-за 53 особенностей реализации обращение к элементам по индексу, которое выполняется за константное время $O(1)$. Использование данной коллекции рекомендуется избегать, если требуется частое удаление/добавление элементов в середине коллекции.

LinkedList — вид реализации List. Позволяет хранить любые данные, включая null. Данная коллекция реализована на основе двунаправленного связного списка (каждый элемент списка имеет ссылки на предыдущий и следующий). Добавление и удаление элемента из середины, доступ по индексу, значению происходит за линейное время $O(n)$, а из начала и конца за константное время $O(1)$. Ввиду реализации, данную коллекцию можно использовать как абстрактный тип данных — стек или очередь. Для этого в ней реализованы соответствующие методы

РЕШЕНИЕ ЗАДАЧИ

1. Постановка задачи

Задание:

Напишите программу в виде консольного приложения, которая моделирует карточную игру «пьяница» и определяет, кто выигрывает. В игре участвует 10 карт, имеющих значения от 0 до 9, большая карта побеждает меньшую; карта «0» побеждает карту «9».

Карточная игра “ В пьяницу”. В этой игре карточная колода раздается поровну двум игрокам. Далее они открывают по одной верхней карте, и тот, чья карта старше, забирает себе обе открытые карты, которые кладутся под низ его колоды. Тот, кто остается без карт, - проигрывает.

Для простоты будем считать, что все карты различны по номиналу и что самая младшая карта побеждает самую старшую карту (“шестерка берет туз”).

Игрок, который забирает себе карты, сначала кладет под низ своей колоды карту первого игрока, затем карту второго игрока (то есть карта второго игрока оказывается внизу колоды).

- 1) Используйте для организации хранения структуру данных Stack.
- 2) Используйте для организации хранения структуру данных Queue.
- 3) Используйте для организации хранения структуру данных Dequeue.
- 4) Используйте для организации хранения структуру данных DoubleList.

Входные данные:

Программа получает на вход две строки: первая строка содержит 5 карт первого игрока, вторая - 5 карт второго игрока. Карты перечислены сверху вниз, то есть каждая строка начинается с той карты, которая будет открыта первой.

Выходные данные:

Программа должна определить, кто выигрывает при данной раздаче, и вывести слово first или second, после чего вывести количество ходов, сделанных до выигрыша. Если на протяжении 106 ходов игра не заканчивается, программа должна вывести слово botva.

2. Программный код

Файл QueueTest.java:

```
package ru.mirea;

import java.util.Scanner;
import java.util.LinkedList;

public class QueueTest {
    private static final int AMOUNT_OF_CARDS = 5;
```

```

private static final Scanner IN = new Scanner(System.in);

public static void main(String[] args) {
    var firstPlayerCards = new LinkedList<Integer>();
    var secondPlayerCards = new LinkedList<Integer>();

    System.out.print("Введите карты 1-го игрока: ");
    for (int i = 0; i < AMOUNT_OF_CARDS; ++i) {
        firstPlayerCards.add(IN.nextInt());
    }

    System.out.print("Введите карты 2-го игрока: ");
    for (int i = 0; i < AMOUNT_OF_CARDS; ++i) {
        secondPlayerCards.add(IN.nextInt());
    }

    int steps = 0;

    while (true) {
        if (steps >= 106) {
            System.out.println();
            System.out.println("botva");
            return;
        }

        if (firstPlayerCards.isEmpty())
            break;

        int firstPlayerCard = firstPlayerCards.poll();

        if (secondPlayerCards.isEmpty())
            break;

        int secondPlayerCard = secondPlayerCards.poll();

        if (firstPlayerCard < secondPlayerCard) {
            if (firstPlayerCard == 0 && secondPlayerCard == 9) { //
                младшая карта побеждает старшую
                firstPlayerCards.add(firstPlayerCard);
                firstPlayerCards.add(secondPlayerCard);
            }
            else {
                secondPlayerCards.add(firstPlayerCard);
                secondPlayerCards.add(secondPlayerCard);
            }
        }
        else {

```

```

        if (firstPlayerCard == 9 && secondPlayerCard == 0) {    //
младшая карта побеждает старшую
            secondPlayerCards.add(firstPlayerCard);
            secondPlayerCards.add(secondPlayerCard);
        }
        else {
            firstPlayerCards.add(firstPlayerCard);
            firstPlayerCards.add(secondPlayerCard);
        }
    }

    ++steps;
}

String winner = (firstPlayerCards.isEmpty()) ? "second" :
"first";

System.out.println();
System.out.println(winner + " " + steps);
}
}

```

Файл StackTest.java:

```

package ru.mirea;

import java.util.Stack;
import java.util.Scanner;
import java.util.Collections;

public class StackTest {
    private static final int AMOUNT_OF_CARDS = 5;
    private static final Scanner IN = new Scanner(System.in);

    private static void reverse(Stack<Integer> s1, Stack<Integer>
s2) {
        Collections.reverse(s1);
        Collections.reverse(s2);
    }

    public static void main(String[] args) {
        var firstPlayerCards = new Stack<Integer>();
        var secondPlayerCards = new Stack<Integer>();

        System.out.print("Введите карты 1-го игрока: ");
        for (int i = 0; i < AMOUNT_OF_CARDS; ++i) {
            firstPlayerCards.push(IN.nextInt());
        }
    }
}

```

```

System.out.print("Введите карты 2-го игрока: ");
for (int i = 0; i < AMOUNT_OF_CARDS; ++i) {
    secondPlayerCards.push(IN.nextInt());
}

int steps = 0;

while (true) {
    if (steps >= 106) {
        System.out.println();
        System.out.println("botva");
        return;
    }

    reverse(firstPlayerCards, secondPlayerCards);

    if (firstPlayerCards.isEmpty())
        break;

    int firstPlayerCard = firstPlayerCards.pop();

    if (secondPlayerCards.isEmpty())
        break;

    int secondPlayerCard = secondPlayerCards.pop();

    reverse(firstPlayerCards, secondPlayerCards);
    if (firstPlayerCard < secondPlayerCard) {
        if (firstPlayerCard == 0 && secondPlayerCard == 9) {    //
младшая карта побеждает старшую
            firstPlayerCards.push(firstPlayerCard);
            firstPlayerCards.push(secondPlayerCard);
        }
        else {
            secondPlayerCards.push(firstPlayerCard);
            secondPlayerCards.push(secondPlayerCard);
        }
    }
    else {
        if (firstPlayerCard == 9 && secondPlayerCard == 0) {    //
младшая карта побеждает старшую
            secondPlayerCards.push(firstPlayerCard);
            secondPlayerCards.push(secondPlayerCard);
        }
        else {
            firstPlayerCards.push(firstPlayerCard);

```

```

        firstPlayerCards.push(secondPlayerCard);
    }
}

++steps;
}

String winner = (firstPlayerCards.isEmpty()) ? "second" :
"first";

System.out.println();
System.out.println(winner + " " + steps);
}
}

```

Файл Deque.java:

```

package ru.mirea;

import java.util.Scanner;
import java.util.ArrayDeque;

public class Deque {
    private static final int AMOUNT_OF_CARDS = 5;
    private static final Scanner IN = new Scanner(System.in);

    public static void main(String[] args) {
        var firstPlayerCards = new ArrayDeque<Integer>();
        var secondPlayerCards = new ArrayDeque<Integer>();

        System.out.print("Введите карты 1-го игрока: ");
        for (int i = 0; i < AMOUNT_OF_CARDS; ++i) {
            firstPlayerCards.add(IN.nextInt());
        }

        System.out.print("Введите карты 2-го игрока: ");
        for (int i = 0; i < AMOUNT_OF_CARDS; ++i) {
            secondPlayerCards.add(IN.nextInt());
        }

        int steps = 0;

        while (true) {
            if (steps >= 106) {
                System.out.println();
                System.out.println("botva");
                return;
            }
        }
    }
}

```



```

        if (firstPlayerCards.isEmpty())
            break;

        int firstPlayerCard = firstPlayerCards.poll();

        if (secondPlayerCards.isEmpty())
            break;

        int secondPlayerCard = secondPlayerCards.poll();

        if (firstPlayerCard < secondPlayerCard) {
            if (firstPlayerCard == 0 && secondPlayerCard == 9) {    //
младшая карта побеждает старшую
                firstPlayerCards.add(firstPlayerCard);
                firstPlayerCards.add(secondPlayerCard);
            }
            else {
                secondPlayerCards.add(firstPlayerCard);
                secondPlayerCards.add(secondPlayerCard);
            }
        }
        else {
            if (firstPlayerCard == 9 && secondPlayerCard == 0) {    //
младшая карта побеждает старшую
                secondPlayerCards.add(firstPlayerCard);
                secondPlayerCards.add(secondPlayerCard);
            }
            else {
                firstPlayerCards.add(firstPlayerCard);
                firstPlayerCards.add(secondPlayerCard);
            }
        }

        ++steps;
    }

    String winner = (firstPlayerCards.isEmpty()) ? "second" :
"first";

    System.out.println();
    System.out.println(winner + " " + steps);
}
}

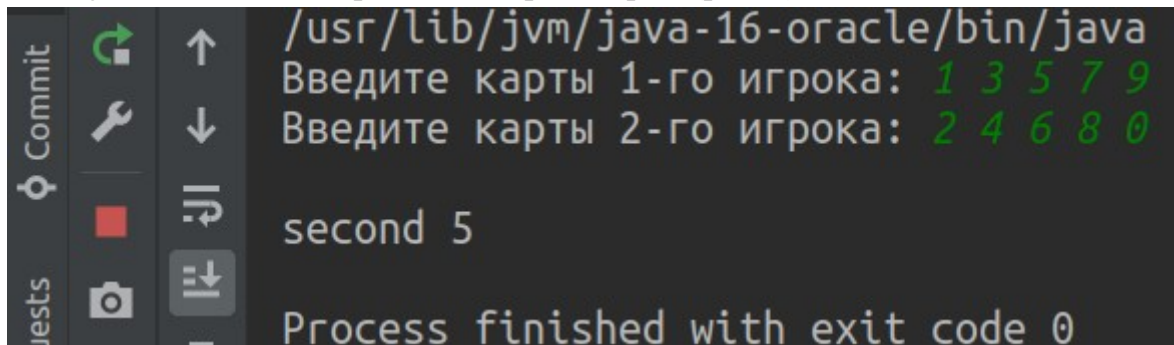
```

В использовании DoubleList нет необходимости, потому что класс LinkedList, который используется в качестве очереди в данном задании,

реализует функционал как для однонаправленного списка, так и для двунаправленного списка.

3. Вывод программы

Случай, когда выигрывает второй игрок (рис.1):



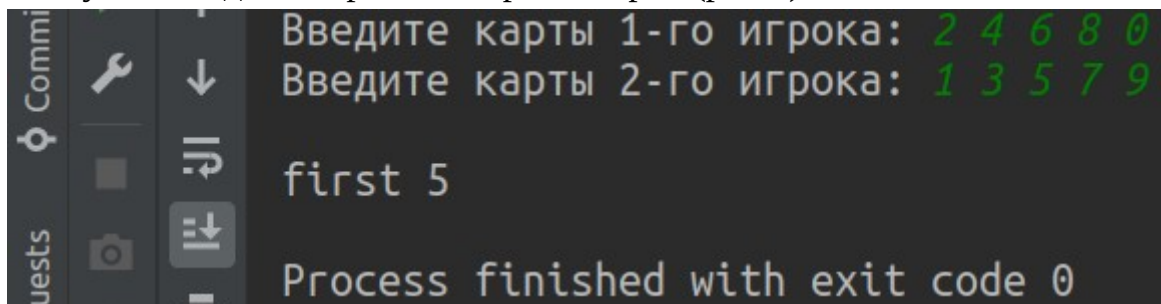
```
/usr/lib/jvm/java-16-oracle/bin/java
Введите карты 1-го игрока: 1 3 5 7 9
Введите карты 2-го игрока: 2 4 6 8 0

second 5

Process finished with exit code 0
```

Рис.1 Вывод программы ч.1

Случай, когда выигрывает первый игрок (рис.2).



```
Введите карты 1-го игрока: 2 4 6 8 0
Введите карты 2-го игрока: 1 3 5 7 9

first 5

Process finished with exit code 0
```

Рис.2 Вывод программы ч.2

ВЫВОД

В ходе выполнения работы изучил на практике приемы работы со стандартными контейнерными классами Java Collection Framework.