|  |  |
| --- | --- |
|  |  |
| МИНОБРНАУКИ РОССИИ |  |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |  |

**Институт информационных технологий**

КАФЕДРА ИНСТРУМЕТНАЛЬНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ИППО)

ПРАКТИЧЕСКАЯ РАБОТА № 7

по дисциплине «Объектно-ориентированное программирование»

«Очереди»

ВАРИАНТ 1

Выполнил студент группы ИНБО-01-18 *Трофимов С. М.*

Принял Батанов А.О.

Практическая работа выполнена «\_\_\_»\_\_\_\_\_\_\_2019г.

«Зачтено» «\_\_\_»\_\_\_\_\_\_\_2019г.

Москва 2019

**Цель работы**

Изучение работы с очередями.

**Теоретическое введение**

Конструкторы:

* Stack() Создаёт пустой стек.

Методы:

* boolean empty() - Служит для проверки стека на наличие элементов — он возвращает true, если стек пуст.
* Object peek() - Возвращает верхний элемент, не удаляя его из стека.
* Object pop() - Извлекает верхний элемент удаляя его из стека.
* Object push(Object item) - Помещает элемент в вершину стека.
* int search(Object o) - Метод ищет заданный элемент в стеке, возвращая количество операций pop, которые требуются для того чтобы перевести искомый элемент в вершину стека. Если заданный элемент в стеке отсутствует, этот метод возвращает -1.

**Выполнение практической работы**

*Задание:*

В игре в пьяницу карточная колода раздается поровну двум игрокам. Далее они вскрывают по одной верхней карте, и тот, чья карта старше, забирает себе обе вскрытые карты, которые кладутся под низ его колоды. Тот, кто остается без карт - проигрывает.

Для простоты будем считать, что все карты различны по номиналу, а также, что самая младшая карта побеждает самую старшую карту ("шестерка берет туза").

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).

Напишите программу, которая моделирует игру в пьяницу и определяет, кто выигрывает. В игре участвует 10 карт, имеющих значения от 0 до 9, большая карта побеждает меньшую, карта со значением 0 побеждает карту 9.

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.

Пример ввода:

1 3 5 7 9

2 4 6 8 0

Вывод:

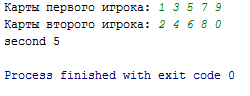
second 5

*Решение:*

Файл Main.java

**package** com.gradess.company;  
  
**import** java.util.\*;  
  
**public class** Main {  
  
 **private static int** *QUEUE\_SIZE* = 5;  
 **private static** ArrayDeque<Integer> *firstPlayer* = **new** ArrayDeque<>(*QUEUE\_SIZE*);  
 **private static** ArrayDeque<Integer> *secondPlayer* = **new** ArrayDeque<>(*QUEUE\_SIZE*);  
  
 **private static** Integer[] *cardsArray* = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};  
 **private static** ArrayList<Integer> *availableCards* = **new** ArrayList<>(Arrays.*asList*(*cardsArray*));  
  
  
 **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
  
 System.***out***.print(**"Карты первого игрока: "**);  
 *fillQueue*(*firstPlayer*, scanner);  
  
 System.***out***.print(**"Карты второго игрока: "**);  
 *fillQueue*(*secondPlayer*, scanner);  
  
 scanner.close();  
  
 *play*();  
 }  
  
 **private static void** fillQueue(Deque<Integer> queue, Scanner scanner) {  
 **for** (**int** i = 0; i < *QUEUE\_SIZE*; i++) {  
 Integer newElement = scanner.nextInt();  
 queue.addLast(newElement);  
 }  
 }  
  
 **private static void** play() {  
 **int** counter = 0;  
 **while** (*canContinueGame*(counter)) {  
  
 Integer firstElem = *firstPlayer*.pollFirst();  
 Integer secondElem = *secondPlayer*.pollFirst();  
  
 **if** (firstElem != **null** && secondElem != **null**) {  
 **if** (*isFirstWinRound*(firstElem, secondElem)) {  
 *firstPlayer*.addLast(firstElem);  
 *firstPlayer*.addLast(secondElem);  
 } **else** {  
 *secondPlayer*.addLast(secondElem);  
 *secondPlayer*.addLast(firstElem);  
 }  
 }  
  
 counter++;  
 }  
  
 **if** (counter > 106) {  
 System.***out***.println(**"botva"**);  
 } **else** {  
 System.***out***.println((*firstPlayer*.isEmpty() ? **"second "** : **"first "** ) + counter);  
 }  
 }  
  
 **private static boolean** canContinueGame(**int** counter) {  
 **return** !*firstPlayer*.isEmpty() && !*secondPlayer*.isEmpty() && counter < 106;  
 }  
  
 **private static boolean** isFirstWinRound(Integer firstElem, Integer secondElem) {  
 **return** ((firstElem != 9 || secondElem != 0) && firstElem > secondElem) || (firstElem == 0 && secondElem == 9);  
 }  
  
 *//For debugging* @SuppressWarnings(**"unused"**)  
 **private static void** randomFillQueue(Deque<Integer> queue) {  
 **for** (**int** i = 0; i < *QUEUE\_SIZE*; i++) {  
 **int** randomId = (**int**) (Math.*random*() \* *availableCards*.size());  
 queue.addLast(*availableCards*.remove(randomId));  
 System.***out***.println(queue.peekLast() + **" "**);  
 }  
 System.***out***.println();  
 }  
}

**Результат выполнения программы**



**Выводы по работе**

Я изучил на практике работу с очередями.

**Используемая литература:**

1. К. Сьерра, Б. Бэйтс. Изучаем Java, 2-е издание.
2. С. Макконнелл. Совершенный код, 3-е издание.