**Задание.**

Выполнить действия по разработке классов с использованием наследования. Протестировать каждый из написанных классов в отдельной процедуре в файле Main.java. Оформить отчет по работе.

Обязательные – первые 2 пункта. По желанию все 4 пункта.

1. Создать класс IntArray – целочисленный массив фиксированного размера.

Поля:

* массивы целых чисел – данные массива

Методы:

* конструкторы
* необходимые геттеры/сеттеры (в т.ч. вычисляемые)
* получение/установка элемента по индексу
* toString

2. Создать класс IntStack – наследник IntArray – целочисленный стек, использующий для хранения элементов массив.

Поля:

* размер стека

Методы:

* конструкторы
* необходимые геттеры/сеттеры (в т.ч. вычисляемые)
* добавление элемента в верхушку стека
* удаление элемента из верхушки стека
* получение элемента с верхушки стека без удаления
* toString
* deepToString

Примечание: можно использовать код с занятия.

3. Создать класс IntQueue – наследник IntArray – целочисленная очередь, использующая для хранения элементов массив.

Поля:

* размер очереди (сколько элементов хранится фактически, размер массива – есть ёмкость очереди)

Методы:

* конструкторы
* необходимые геттеры/сеттеры (в т.ч. вычисляемые)
* добавление элемента в очередь (принцип FIFO)
* удаление элемента из очереди (принцип FIFO)
* toString
* deepToString – вывод не только элементов очереди, а текущего состояние всего массива данных очереди (вызов toString у базового класса IntArray)

Примечание: будет очень хорошо, если алгоритмы добавления/удаления элементов из очереди не потребуют сдвига элементов массива, в котором хранятся данные (необходимо «играться» с индексами начала и конца очереди).

4. Создать класс IntDeque – наследник IntStack – целочисленный дек, использующий для хранения элементов массив и реализующий поведение обычного стека.

Поля:

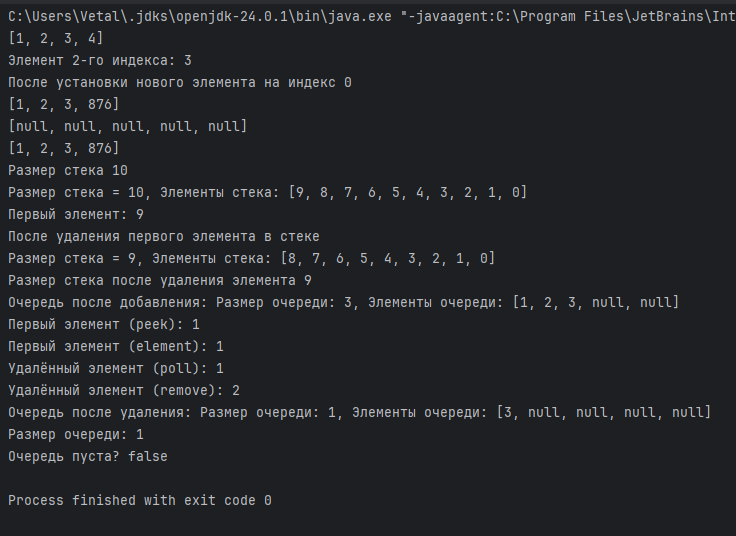
* в случае использования рационального по времени алгоритма добавления/удаления необходимые индексы начала и конца дека

Методы:

* конструкторы
* необходимые геттеры/сеттеры (в т.ч. вычисляемые)
* добавление элемента в начало дека (pushFront)
* удаление элемента из начала дека (popFront)
* получение элемента из начала дека без удаления (topFront)
* toString
* deepToString

Примечание: будет очень хорошо, если алгоритмы добавления/удаления элементов в начале дека не потребуют сдвига элементов массива, в котором хранятся данные (необходимо «играться» с индексами начала и конца дека).

Выполнение



Код Main

package org.example;  
  
import org.example.classes.IntArray;  
import org.example.classes.IntQueue;  
import org.example.classes.IntStack;  
  
public class Main {  
 public static void main(String[] args)  
 {  
  
 //класс IntArray  
 Integer[] value = {1, 2, 3, 4};  
 IntArray customArr = new IntArray(value);  
 System.*out*.println(customArr.toString());  
 System.*out*.println("Элемент 2-го индекса: " + customArr.getElement(2));  
 customArr.setElement(3, 876);  
 System.*out*.println("После установки нового элемента на индекс 0");  
 System.*out*.println(customArr.toString());  
 IntArray customArr2 = new IntArray(5);  
 System.*out*.println(customArr2.toString());  
 customArr2.setArray(customArr.getArray());  
 System.*out*.println(customArr2.toString());  
  
 //класс IntStack  
 IntStack customStack = new IntStack();  
 for (int i = 0; i < 10; i++)  
 {  
 customStack.push(i);  
 }  
 System.*out*.println("Размер стека "+customStack.getStackSize());  
 System.*out*.println(customStack.toString());  
 System.*out*.println("Первый элемент: " + customStack.peek());  
 customStack.pop();  
 System.*out*.println("После удаления первого элемента в стеке");  
 System.*out*.println(customStack.toString());  
 System.*out*.println("Размер стека после удаления элемента "+customStack.getStackSize());  
  
 //класс IntQueue  
 IntQueue customQueue = new IntQueue(5);  
  
 customQueue.offer(1);  
 customQueue.offer(2);  
 customQueue.offer(3);  
 System.*out*.println("Очередь после добавления: " + customQueue);  
 System.*out*.println("Первый элемент (peek): " + customQueue.peek());  
 System.*out*.println("Первый элемент (element): " + customQueue.element());  
 System.*out*.println("Удалённый элемент (poll): " + customQueue.poll());  
 System.*out*.println("Удалённый элемент (remove): " + customQueue.remove());  
 System.*out*.println("Очередь после удаления: " + customQueue);  
 System.*out*.println("Размер очереди: " + customQueue.size());  
 System.*out*.println("Очередь пуста? " + customQueue.isEmpty());  
  
 }  
}

Код IntArray

package org.example.classes;  
  
import java.util.Arrays;  
  
public class IntArray  
{  
 public Integer[] array;  
  
 public IntArray(Integer[] array)  
 {  
 this.array = array;  
 }  
  
 public IntArray(int size)  
 {  
 this.array = new Integer[size];  
 }  
  
 public void setArray(Integer[] array)  
 {  
 this.array = array;  
 }  
  
 public Integer[] getArray()  
 {  
 return array;  
 }  
  
 public Integer getElement(int index)  
 {  
 if (index < 0 || index >= array.length)  
 {  
 System.*out*.println("Индекс вне диапазона массива");  
 return null;  
 }  
 return this.array[index];  
 }  
  
 public void setElement(int index, int element)  
 {  
 if (index < 0 || index >= array.length)  
 {  
 System.*out*.println("Индекс вне диапазона массива");  
 return;  
 }  
 this.array[index] = element;  
 }  
  
 @Override  
 public String toString()  
 {  
 return Arrays.*toString*(array);  
 }  
}

Код IntStack

package org.example.classes;  
  
public class IntStack extends IntArray  
{  
 public int stackSize;  
  
 public IntStack() {  
 super(0);  
 stackSize = 0;  
 }  
  
 public int getStackSize()  
 {  
 return stackSize;  
 }  
  
 public void push(int element)  
 {  
 Integer[] temp = new Integer[array.length + 1];  
 temp[0] = element;  
 System.*arraycopy*(array, 0, temp, 1, array.length);  
 array = temp;  
 stackSize ++;  
 }  
  
 public boolean isEmpty()  
 {  
 return !(array.length>0);  
 }  
  
 public Integer peek()  
 {  
 if(isEmpty())  
 {  
 throw new IllegalArgumentException("Стек пуст");  
 }  
 return getElement(0);  
 }  
  
  
 public int pop()  
 {  
 if(isEmpty())  
 {  
 throw new IllegalArgumentException("Стек пуст");  
 }  
 int first = this.peek();  
 Integer[] temp = new Integer[array.length - 1];  
 System.*arraycopy*(array, 1, temp, 0, array.length - 1);  
 array = temp;  
 stackSize--;  
 return first;  
 }  
  
 @Override  
 public String toString(){  
 return "Размер стека = "+stackSize+", "+ "Элементы стека: " + super.toString();  
 }  
  
}

Код IntQueue

package org.example.classes;  
import java.util.NoSuchElementException;  
  
public class IntQueue extends IntArray  
{  
 private int size;  
  
 public IntQueue()  
 {  
 this(0);  
 }  
  
 public IntQueue(int capacity)  
 {  
 super(capacity);  
 this.size = 0;  
 }  
  
 public void add(int value)  
 {  
 if (size == array.length)  
 {  
 throw new IllegalStateException("Очередь полна");  
 }  
 array[size] = value;  
 size++;  
 }  
  
 public boolean offer(int value)  
 {  
 if (size == array.length)  
 {  
 return false;  
 }  
 else  
 {  
 this.add(value);  
 return true;  
 }  
 }  
  
 public int remove()  
 {  
 if (size == 0)  
 {  
 throw new NoSuchElementException("Очередь пуста");  
 }  
 int value = array[0];  
 // Сдвиг всех элементов на одну позицию влево  
 System.*arraycopy*(array, 1, array, 0, size - 1);  
 array[size - 1] = null; // Очистка пустой ячейки  
 size--;  
 return value;  
 }  
  
 public Integer poll()  
 {  
 if (size == 0)  
 {  
 return null;  
 }  
 return remove();  
 }  
  
  
 public int element()  
 {  
 if (size == 0)  
 {  
 throw new NoSuchElementException("Очередь пуста");  
 }  
 return array[0];  
 }  
  
 public Integer peek()  
 {  
 if (size == 0)  
 {  
 return null;  
 }  
 return element();  
 }  
  
 public int size()  
 {  
 return size;  
 }  
  
 public boolean isEmpty()  
 {  
 return size == 0;  
 }  
  
 public boolean isFull()  
 {  
 return size == array.length;  
 }  
  
 @Override  
 public String toString() {  
 return "Размер очереди: " + size() + ", Элементы очереди: " + super.toString();  
 }  
}