ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»**

**(НИУ «БелГУ»)**

**ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ**

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

**ОТЧЕТ**

**по лабораторной работе №4**

студента очной формы обучения 3 курса группы 12001905

Инченко Виктор Владиславович

Проверил:

Ст. преподаватель

Лифиренко М.В.

**Оценка*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г.

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись (расшифровка подписи)

Белгород 2022

**Цель работы:** получить практические навыки создания многопоточных приложений.

(Вариант №4)

**Листинг класса Main**

package com.company;  
import java.util.ArrayList;  
import java.util.List;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Port port = new Port(4, 5000, 1000);  
  
 List<Ship> ships = new ArrayList<>();  
  
 for (int i = 0; i < 4; i++) {  
 ships.add(new Ship("Ship " + i, 260, 0, port));  
 }  
  
 try {  
 Thread.*sleep*(3500);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 for (int i = 4; i < 8; i++) {  
 ships.add(new Ship("Ship " + i, 0, 300, port));  
 }  
  
 for (Ship ship : ships){  
 try {  
 ship.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 System.*out*.println("All ships have finished their task");  
 }  
}

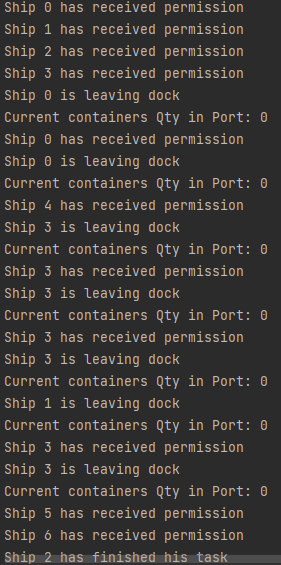
**Листинг класса Ship**

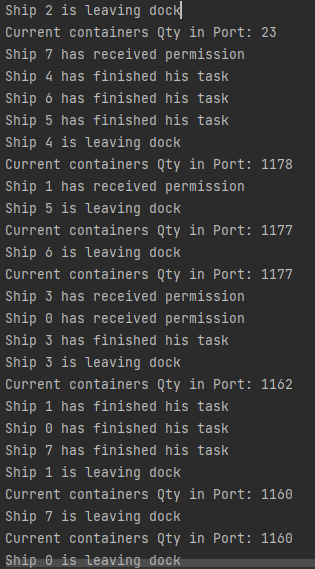
package com.company;  
  
public class Ship extends Thread {  
 int containersToTake;  
 int containersToLeave;  
 Port port;  
  
 int currentContainersQty;  
  
 public Ship(String name, int containersToTake, int containersToLeave, Port port) {  
 super(name);  
 this.containersToTake = containersToTake;  
 this.containersToLeave = containersToLeave;  
 this.port = port;  
 currentContainersQty = containersToLeave;  
 start();  
 }  
  
 @Override  
 public void run() {  
 boolean isChanged = false;  
  
 try {  
  
 while (true) {  
 if (!isChanged) {  
 port.askPermission();  
 }  
  
 isChanged = false;  
 if (containersToLeave != 0 && containersToTake != 0) {  
 containersToTake--;  
 containersToLeave--;  
 isChanged = true;  
 } else {  
 if (containersToLeave != 0 ){  
 synchronized (port) {  
 if (port.getContainersCapacity() > port.getCurrentContainersQty()) {  
 port.addContainer();  
 containersToLeave--;  
 isChanged = true;  
 }  
 }  
 } else {  
 if (containersToTake != 0 ){  
 synchronized (port) {  
 if (port.getCurrentContainersQty() > 0) {  
 port.takeConatiner();  
 containersToTake--;  
 isChanged = true;  
 }  
 }  
 } else {  
 System.*out*.println(Thread.*currentThread*().getName() + " has finished his task");  
 port.returnPermission();  
 break;  
 }  
 }  
 }  
  
 if (isChanged){  
 Thread.*sleep*(10);  
 } else {  
 port.returnPermission();  
 }  
 }  
 }catch (InterruptedException e){  
 e.printStackTrace();  
 }  
  
 }  
  
}

**Листинг класса Port**

package com.company;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Port {  
 private int dockQty;  
 private int containersCapacity;  
 private int currentContainersQty;  
  
 List<Thread> ships = new ArrayList<>();  
  
 public Port(int dockQty, int containersCapacity, int currentContainersQty) {  
 this.dockQty = dockQty;  
 this.containersCapacity = containersCapacity;  
 this.currentContainersQty = currentContainersQty;  
 }  
  
 public int getContainersCapacity() {  
 return containersCapacity;  
 }  
  
 public int getCurrentContainersQty() {  
 return currentContainersQty;  
 }  
  
 public void addContainer() {  
 currentContainersQty++;  
 }  
 public void takeConatiner() {  
 currentContainersQty--;  
 }  
  
  
 public synchronized void askPermission(){  
 while (dockQty == 0){  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 ships.add(Thread.*currentThread*());  
 System.*out*.println(Thread.*currentThread*().getName() + " has received permission");  
 dockQty--;  
 }  
  
 public synchronized void returnPermission() {  
// Time needed to go out from dock for ship  
 try {  
 Thread.*sleep*(100);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 System.*out*.println(Thread.*currentThread*().getName() + " is leaving dock");  
  
 System.*out*.println("Current containers Qty in Port: " + currentContainersQty);  
  
 if (ships.contains(Thread.*currentThread*())) {  
 dockQty++;  
 }  
 ships.remove(Thread.*currentThread*());  
  
 notifyAll();  
 }  
}

Результаты работы программы представлен на рисунке 1.





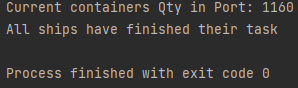


Рисунок 1 – Результат работы программы

Вывод: были получены практические навыки создания многопоточных приложений.