1. Порт

Корабли заходят в порт для разгрузки/загрузки контейнеров. Число контейнеров, находящихся в текущий момент в порту и на корабле, должно быть неотрицательным и превышающим заданную грузоподъемность судна и вместимость порта. В порту работает несколько причалов. У одного причала может стоять один корабль. Корабль может загружаться у причала, разгружаться или выполнять оба действия.

Main.java

package eSForgary;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Main {  
 public static void main(String[] args) {  
 Port port = new Port(4, 3000, 800);  
 List<Ship> ships = new ArrayList<>();  
  
 for (int i = 0; i < 4; i++) {  
 ships.add(new Ship("Ship " + (i + 1), 260, 0, port));  
 }  
  
 try {  
 Thread.*sleep*(3500);  
 } catch (InterruptedException exception) {  
 exception.printStackTrace();  
 }  
  
 for (int i = 4; i < 8; i++) {  
 ships.add(new Ship("Ship " + i, 0, 500, port));  
 }  
  
 for (Ship ship : ships) {  
 try {  
 ship.join();  
 } catch (InterruptedException exception) {  
 exception.printStackTrace();  
 }  
 }  
 System.*out*.println("All ships have finished their task");  
 }  
}

Ship.java

package eSForgary;  
  
public class Ship extends Thread {  
 int containersToTake;  
 int containersToLeave;  
 Port port;  
 int currentContainersCount;  
  
 public Ship(String name, int containersToTake, int containersToLeave, Port port) {  
 super(name);  
 this.containersToTake = containersToTake;  
 this.containersToLeave = containersToLeave;  
 this.port = port;  
 currentContainersCount = 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.getCurrentContainersCount()) {  
 port.addContainer();  
 containersToLeave--;  
 isChanged = true;  
 }  
 }  
 } else {  
 if (containersToTake != 0 ) {  
 synchronized (port) {  
 if (port.getCurrentContainersCount() > 0) {  
 port.takeContainer();  
 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 exception) {  
 exception.printStackTrace();  
 }  
 }  
}

Port.java

package eSForgary;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Port {  
 private int dockCount;  
 private int containersCapacity;  
 private int currentContainersCount;  
 List<Thread> ships = new ArrayList<>();  
  
 public Port(int dockQty, int containersCapacity, int currentContainersQty) {  
 this.dockCount = dockQty;  
 this.containersCapacity = containersCapacity;  
 this.currentContainersCount = currentContainersQty;  
 }  
  
 public int getContainersCapacity() { return containersCapacity; }  
 public int getCurrentContainersCount() { return currentContainersCount; }  
 public void addContainer() { currentContainersCount++; }  
 public void takeContainer() { currentContainersCount--; }  
  
 public synchronized void askPermission() {  
 while (dockCount == 0) {  
 try {  
 wait();  
 } catch (InterruptedException exception) {  
 exception.printStackTrace();  
 }  
 }  
 ships.add(Thread.*currentThread*());  
 System.*out*.println(Thread.*currentThread*().getName() + " has received permission");  
 dockCount--;  
 }  
  
 public synchronized void returnPermission() {  
 try {  
 Thread.*sleep*(100);  
 } catch (InterruptedException exception) {  
 exception.printStackTrace();  
 }  
  
 System.*out*.println(Thread.*currentThread*().getName() + " is leaving dock");  
 System.*out*.println("Current containers Qty in Port: " + currentContainersCount);  
  
 if (ships.contains(Thread.*currentThread*())) {  
 dockCount++;  
 }  
 ships.remove(Thread.*currentThread*());  
  
 notifyAll();  
 }  
}

