**WOJSKOWA AKADEMIA TECHNICZNA**Im. Jarosława Dąbrowskiego

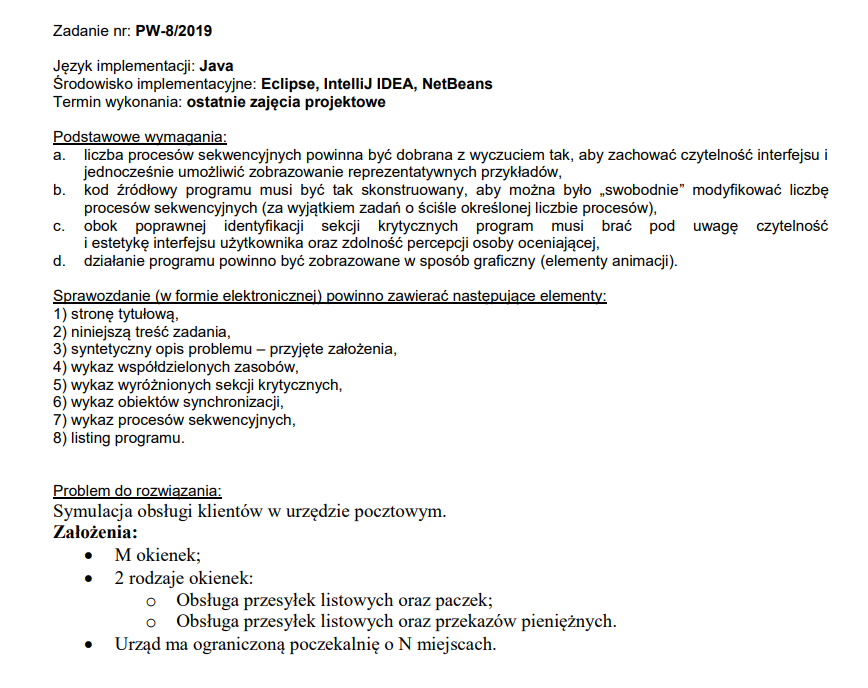
**WYDZIAŁ CYBERNETYKI**

**Projekt**

***Programowanie współbieżne   
2019***

*Autor: Paweł Kierski   
Grupa: I7Y2S1*

*Prowadzący: dr inż. Jarosław Rulka*



Jak wskazuje treść zadania problemem postawionym przy realizacji projektu jest jednoczesna obsługa wielu klientów urzędu pocztowego. W rzeczywistości wygląda to tak, iż interesant przychodząc do urzędu odbiera numerek z automatu postawionego przy wejściu, po czym oczekuje, aż na wyświetlaczu nad okienkiem pojawi się numer pasujący do tego który został mu przydzielony. Pozwala to na bezkonfliktową obsługę dużej liczby klientów. Mamy pewność, że żaden z nich niesprawiedliwie nie „wciśnie” się przed kogoś kto był przed nim. Dodatkowo osoba zainteresowana dokładnie wie kiedy jej kolej więc oszczędzamy zbędne pytania w stronę osób pracujących w urzędzie pocztowym.

W postawionym zadaniu urząd pozwala na załatwienie spraw zawierających się w kategoriach; listy, paczki oraz przekazy pieniężne. Każdy klient więc został przedstawiony symbolicznie jako koło; gdzie sprawę z którą przychodzi reprezentuje kolor. Analogicznie okienka w urzędzie mają ten sam zestaw kolorów. Ponieważ w okienkach obu typu możemy załatwić sprawę związaną z listami każde okienko ma czarną ramkę.



Postawiony problem został rozwiązany z pomocą Algorytmu Producenci i Konsumenci. Producentami są klienci – którzy „produkowali” instancję klasy Klient na współdzielonej liście reprezentującej kolejkę w poczekalni. Obiekty – okienka z klasy Okienka są natomiast konsumentami, które w zależności od swoich kompetencji (typ sprawy) pobierają rekord Klienta z listy o najniższym numerze (FIFO). Typ sprawy u producentów oraz kompetencje u konsumentów przydzielane są w sposób losowy.

Do zaimplementowania algorytmu został wykorzystany prosty monitor z dwiema zmiennymi warunkowymi, mówiącymi o stanie kolejki (pusta, pełna) oraz semafor, który „wspiera” monitor.

public void stanWKolejce(Klient klient) throws InterruptedException {  
  
 dostep.lock();  
 while (!stanPoczekalni.tryAcquire()) {  
 dostep.unlock();  
 synchronized (poczekalniaPelna) {  
 poczekalniaPelna.wait();  
 }  
 dostep.lock();  
 }  
 *kolejka*.add(klient);  
 klient.setPozycjawKolejce(*kolejka*.indexOf(klient));  
 synchronized (poczekalniaPusta) {  
 poczekalniaPusta.notify();  
 }  
 Thread.*sleep*(Klient.*move*(1300 - klient.getPozycjawKolejce() \* 2.5 \* klient.getRadius(), 800, klient) / 15);  
 dostep.unlock();  
}

}

*Fragment kodu monitora dodającym Klienta do kolejki*

public Klient poczatekObslugi(Okienko okienko) throws InterruptedException {  
  
 Klient obslugiwanyKlient;  
 dostep.lock();  
 while (*czyOdpowiednieOkienko*(okienko) == null) {  
 dostep.unlock();  
 synchronized (poczekalniaPusta) {  
 poczekalniaPusta.wait();  
 }  
 dostep.lock();  
 }  
 obslugiwanyKlient = *czyOdpowiednieOkienko*(okienko);  
 okienko.obsluguje = true;  
 *kolejka*.remove(obslugiwanyKlient);  
 Klient.*move*(okienko.getLayoutX() + okienko.getWidth() / 2, okienko.getLayoutY() + okienko.getHeight() + 10, obslugiwanyKlient);  
 Thread.*sleep*(100);  
 naprawKolejke();  
 stanPoczekalni.release();  
 synchronized (poczekalniaPelna) {  
 poczekalniaPelna.notify();  
 }  
 dostep.unlock();  
 return obslugiwanyKlient;  
}

*Fragment kodu odpowiadający za animacje podejścia klienta do okienka*

public Klient koniecObslugi(Klient obslugiwanyKlient, Okienko okienko) {  
  
 dostep.lock();  
 try {  
 Thread.*sleep*(Klient.*move*(obslugiwanyKlient.getTranslateX(), obslugiwanyKlient.getTranslateY() + 50, obslugiwanyKlient));  
 Klient.*move*(-20, obslugiwanyKlient.getTranslateY(), obslugiwanyKlient);  
 okienko.obsluguje = false;  
 *obsluzono* = *obsluzono* + 1;  
 Poczta.*text*.setText("Klienci obsłużeni: " + *obsluzono*.toString());  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 dostep.unlock();  
 return obslugiwanyKlient;  
}

*Fragment kodu odpowiadający za odejście klienta od okienka*

**Listing całościowy**

package sample;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class Main extends Application {  
  
 @Override  
 public void start(Stage primaryStage) throws Exception{  
 Parent root = FXMLLoader.*load*(getClass().getResource("sample.fxml"));  
 primaryStage.setTitle("Urzad pocztowy");  
 primaryStage.setScene(new Scene(root, 1440, 900));  
 primaryStage.show();  
 }  
  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

package sample;  
import javafx.fxml.FXML;  
import javafx.fxml.Initializable;  
import javafx.scene.control.Button;  
import javafx.scene.control.CheckBox;  
import javafx.scene.control.ChoiceBox;  
import javafx.scene.layout.Pane;  
import javafx.scene.text.Text;  
import java.net.URL;  
import java.util.ResourceBundle;  
public class Poczta implements Initializable {  
  
 @FXML  
 private Pane mainPane;  
 @FXML  
 private ChoiceBox<Integer> liczbaMiejscPoczekalnia;  
 @FXML  
 private ChoiceBox<Integer> liczbaOkienek;  
 @FXML  
 private ChoiceBox<Integer> liczbaKlientow;  
  
 public static Text *text* = new Text();  
  
 static int *M*; //liczba okienek  
 static int *N*; //liczba klientow ktorzy w sumie maja odwiedzic poczte  
 static int *O*;  
  
 static Klient[] *klient*;  
 static Okienko[] *okienko*;  
  
 public void initialize(URL url, ResourceBundle resourceBundle) {  
  
 for(int i = 10; i <= 200; i = i + 10){  
 liczbaKlientow.getItems().add(i);  
 }  
 liczbaKlientow.setValue(40);  
  
 for(int i = 1; i <= 10; i++){  
 liczbaOkienek.getItems().add(i);  
 }  
 liczbaOkienek.setValue(5);  
  
 for(int i = 1; i <= 25; i++){  
 liczbaMiejscPoczekalnia.getItems().add(i);  
 }  
 liczbaMiejscPoczekalnia.setValue(10);  
  
 }  
  
 //metode wywoluje ten maly przycisk start w interfejsie  
  
 public void start() throws InterruptedException{  
  
 mainPane.getChildren().add(*text*);  
  
  
  
 *N* = liczbaKlientow.getValue();  
 *M* = liczbaOkienek.getValue();  
 *O* = liczbaMiejscPoczekalnia.getValue();  
  
 Poczekalnia poczekalnia = new Poczekalnia(*O*);  
 *text*.setText("Klienci obsłużeni: " + Poczekalnia.*obsluzono*.toString());  
 *text*.setX(670);  
 *text*.setY(250);  
 *text*.setScaleX(2);  
  
 *klient* = new Klient[*N*];  
 *okienko* = new Okienko[*M*];  
  
  
 for(int i = 0; i < *N*; i++) {  
 mainPane.getChildren().add(*klient*[i] = new Klient(poczekalnia));  
 Thread thread = new Thread(*klient*[i]);  
 thread.setName("interesant " + i);  
 thread.start();  
 }  
  
 for(int i = 0; i < *M*; i++){  
 *okienko*[i] = new Okienko(i, poczekalnia);  
 mainPane.getChildren().add(*okienko*[i].ramka);  
 mainPane.getChildren().add(*okienko*[i]);  
 Thread thread = new Thread(*okienko*[i]);  
 thread.setName("okienko " + i);  
 thread.start();  
 }  
  
 System.*out*.println("Wątki postawione");  
 }

package sample;  
import java.util.ArrayList;  
import java.util.concurrent.Semaphore;  
import java.util.concurrent.locks.Lock;  
import java.util.concurrent.locks.ReentrantLock;  
  
public class Poczekalnia {  
  
 private static int *maxPoczekalni*;  
 private Lock dostep = new ReentrantLock();  
 private Object poczekalniaPusta = new Object();  
 private Object poczekalniaPelna = new Object();  
 private Semaphore stanPoczekalni;  
 public static volatile Integer *obsluzono* = new Integer(0);  
  
  
 private static ArrayList<Klient> *kolejka* = new ArrayList<>();  
  
 Poczekalnia(int liczbaMiejscPoczekalnia) {  
 *maxPoczekalni* = liczbaMiejscPoczekalnia;  
 stanPoczekalni = new Semaphore(*maxPoczekalni*);  
 }  
  
 public void stanWKolejce(Klient klient) throws InterruptedException {  
  
 dostep.lock();  
 while (!stanPoczekalni.tryAcquire()) {  
 dostep.unlock();  
 synchronized (poczekalniaPelna) {  
 poczekalniaPelna.wait();  
 }  
 dostep.lock();  
 }  
 *kolejka*.add(klient);  
 klient.setPozycjawKolejce(*kolejka*.indexOf(klient));  
 synchronized (poczekalniaPusta) {  
 poczekalniaPusta.notify();  
 }  
 Thread.*sleep*(Klient.*move*(1300 - klient.getPozycjawKolejce() \* 2.5 \* klient.getRadius(), 800, klient) / 15);  
 dostep.unlock();  
 }  
  
 public Klient poczatekObslugi(Okienko okienko) throws InterruptedException {  
  
 Klient obslugiwanyKlient;  
 dostep.lock();  
 while (*czyOdpowiednieOkienko*(okienko) == null) {  
 dostep.unlock();  
 synchronized (poczekalniaPusta) {  
 poczekalniaPusta.wait();  
 }  
 dostep.lock();  
 }  
 obslugiwanyKlient = *czyOdpowiednieOkienko*(okienko);  
 okienko.obsluguje = true;  
 *kolejka*.remove(obslugiwanyKlient);  
 Klient.*move*(okienko.getLayoutX() + okienko.getWidth() / 2, okienko.getLayoutY() + okienko.getHeight() + 10, obslugiwanyKlient);  
 Thread.*sleep*(100);  
 naprawKolejke();  
 stanPoczekalni.release();  
 synchronized (poczekalniaPelna) {  
 poczekalniaPelna.notify();  
 }  
 dostep.unlock();  
 return obslugiwanyKlient;  
 }  
  
 public Klient koniecObslugi(Klient obslugiwanyKlient, Okienko okienko) {  
  
 dostep.lock();  
 try {  
 Thread.*sleep*(Klient.*move*(obslugiwanyKlient.getTranslateX(), obslugiwanyKlient.getTranslateY() + 50, obslugiwanyKlient));  
 Klient.*move*(-20, obslugiwanyKlient.getTranslateY(), obslugiwanyKlient);  
 okienko.obsluguje = false;  
 *obsluzono* = *obsluzono* + 1;  
 Poczta.*text*.setText("Klienci obsłużeni: " + *obsluzono*.toString());  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 dostep.unlock();  
 return obslugiwanyKlient;  
 }

static public Klient czyOdpowiednieOkienko(Okienko okienko) {  
 if (*kolejka*.isEmpty())  
 return null;  
 for (Klient e : *kolejka*) {  
 if (e.interes == okienko.typObslugi || e.interes == 0)  
 return e;  
 }  
 return null;  
 }  
  
 public void naprawKolejke() {  
 try {  
 for (Klient e : *kolejka*) {  
 e.setPozycjawKolejce(*kolejka*.indexOf(e));  
 Thread.*sleep*(Klient.*move*(1300 - 2.5 \* e.getRadius() \* e.getPozycjawKolejce(), 800, e) / 20);  
 }  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }

package sample;  
  
import javafx.animation.PathTransition;  
import javafx.application.Platform;  
import javafx.scene.paint.Color;  
import javafx.scene.shape.Circle;  
import javafx.scene.shape.Line;  
import javafx.util.Duration;  
  
import java.util.Random;  
import java.util.concurrent.locks.Lock;  
import java.util.concurrent.locks.ReentrantLock;  
  
public class Klient extends Circle implements Runnable {  
  
  
 //interes; 0 - przesylka listowa; 1 - paczka; 2 - przekaz pieniezny;  
  
 static double *speed* = 2;  
 int interes;  
 private Poczekalnia poczekalnia;  
 private int pozycjawKolejce;  
 static private Random *interesRandom* = new Random();  
  
 Klient(Poczekalnia poczekalnia) {  
  
 setTranslateX(-20);  
 setTranslateY(800);  
 setRadius(20);  
 this.poczekalnia = poczekalnia;  
 interes = *interesRandom*.nextInt(3);  
  
 switch (interes) {  
 case 0:  
 setFill(Color.*BLACK*);  
 break;  
 case 1:  
 setFill(Color.*RED*);  
 break;  
 case 2:  
 setFill(Color.*GREEN*);  
 break;  
 }  
  
 }  
  
 @Override  
 public void run(){  
  
 try {  
 Thread.*sleep*(50);  
 poczekalnia.stanWKolejce(this);  
  
 //Thread.sleep(1000);  
  
 }catch(InterruptedException e){  
 e.printStackTrace();  
 }  
 }  
  
 public void setPozycjawKolejce(int pozycjawKolejce) {  
 this.pozycjawKolejce = pozycjawKolejce;  
 }  
  
 public int getPozycjawKolejce() {  
 return pozycjawKolejce;  
 }  
  
 public static long move(double toX, double toY, Klient klient){  
  
 double dlugoscWektora = Math.*sqrt*(Math.*pow*((toX - klient.getTranslateX()), 2) + Math.*pow*((toY - klient.getTranslateY()), 2));  
 Line path = new Line(klient.getTranslateX(), klient.getTranslateY(), toX, toY);  
 PathTransition animacja = new PathTransition();  
 animacja.setDuration(Duration.*millis*(*speed*\*dlugoscWektora));  
 animacja.setNode(klient);  
 animacja.setPath(path);  
 Platform.*runLater*(animacja::play);  
  
 return (long)(2\*dlugoscWektora);  
 }  
}

package sample;  
  
import javafx.scene.paint.Color;  
import javafx.scene.shape.Rectangle;  
  
import java.util.Random;  
import java.util.concurrent.Semaphore;  
  
  
public class Okienko extends Rectangle implements Runnable {  
  
  
 private Poczekalnia poczekalnia;  
 public int typObslugi;  
 public int numerOkienka;  
 boolean obsluguje;  
 static private Random *randomTypObslugi* = new Random();  
 Semaphore wolne = new Semaphore(0);  
  
 Rectangle ramka;  
  
 Okienko(int numerOkienka, Poczekalnia poczekalnia){  
  
 this.poczekalnia = poczekalnia;  
 ramka = new Rectangle();  
 this.numerOkienka = numerOkienka;  
 typObslugi = (*randomTypObslugi*.nextInt(100))%2 + 1;  
 wolne.release();  
 setWidth(80);  
 setHeight(50);  
  
 ramka.setHeight(getHeight() + getHeight()/4);  
 ramka.setWidth(getWidth() + getHeight()/4);  
 ramka.setFill(Color.*BLACK*);  
  
 switch(this.typObslugi){  
 case 1:  
 setFill(Color.*RED*);  
 break;  
 case 2:  
 setFill(Color.*GREEN*);  
 break;  
 }  
  
 if(numerOkienka == 0){  
 setLayoutX(1200);  
 setLayoutY(300);  
 }  
 else  
 {  
 setLayoutX(Poczta.*okienko*[numerOkienka - 1].getLayoutX() - 1.5 \* getWidth());  
 setLayoutY(Poczta.*okienko*[numerOkienka - 1].getLayoutY());  
 }  
  
 ramka.setLayoutX(getLayoutX() - getHeight()/8);  
 ramka.setLayoutY(getLayoutY() - getHeight()/8);  
 }  
  
 @Override  
 public void run() {  
  
 Klient obslugiwanyKlient;  
 while(true){  
 try{  
 Thread.*sleep*(100);  
 obslugiwanyKlient = poczekalnia.poczatekObslugi(this);  
 Thread.*sleep*(new Random().nextInt(4000) + 1000);  
 poczekalnia.koniecObslugi(obslugiwanyKlient, this);  
 }catch(InterruptedException e){  
 e.printStackTrace();  
 }  
 }  
 }  
}