WOJSKOWA AKADEMIA TECHNICZNA

Sprawozdanie z przedmiotu:

Programowanie Współbieżne

Temat: Symulacja obsługi na stacji paliw

Prowadzący: *dr inż. Jarosław Rulka*

Autor: Mateusz Jasiński WCY20IY2S1

1. Zadanie

Zadanie nr: **PW-3/2021**
Język implementacji: **Java**
Środowisko implementacyjne: **Eclipse, Intelij IDEA, Netbeans**
Termin wykonania: **ostatnie zajęcia**
Podstawowe wymagania:
a. liczba procesów sekwencyjnych powinna być dobrana z wyczuciem tak, aby zachować czytelność interfejsu i jednocześnie umożliwić zobrazowanie reprezentatywnych przykładów,
b. kod źródłowy programu musi być tak skonstruowany, aby można było „swobodnie" modyfikować liczbę procesów sekwencyjnych (za wyjątkiem zadań o ściśle określonej liczbie procesów),
c. graficzne zobrazowanie działania procesów współbieżnych,
d. odczyt domyślnych danych wejściowych ze sformatowanego, tekstowego pliku danych (xml, properties, inne),
e. możliwość modyfikacji danych wejściowych poprzez GUI.
Sprawozdanie (w formie elektronicznej) powinno zawierać następujące elementy:
1) stronę tytułową,
2) niniejszą treść zadania,
3) syntetyczny opis problemu – w tym wszystkie przyjęte założenia,
4) wykaz współdzielonych zasobów,
5) wykaz wyróżnionych punktów synchronizacji,
6) wykaz obiektów synchronizacji,
7) wykaz procesów sekwencyjnych,
8) listing programu.
Problem do rozwiązania:
Symulacja obsługi na stacji paliw.
Założenia:

- Różne rodzaje paliw.
- Zamykanie stacji na pewien czas (uzupełnienia zapasów).
- M – liczba stanowisk z dystrybutorami (na stanowisku może być kilka różnych
- dystrybutorów).
- K – liczba kas.
- N – maksymalna liczba samochodów, które mogą przebywać na stacji.

2. Syntetyczny opis problemu – przyjęte założenia.

Na stacji jest ograniczona liczba stanowisk z dystrybutorami, jak również ograniczona liczba kas. Ponad to maksymalna liczba samochodów przebywających na stacji jest ustalona.
Problem polega na przypisaniu wolnego stanowiska samochodowi w kolejce, danie mu możliwości pobrania paliwa, a następnie przydzielenie mu wolnej kasy, zwolnienie kasy i zwolnienie stanowiska.
Zapasy paliwa na stacji są ograniczone, przez co są przewidziane przerwy w funkcjonowaniu dystrybutorów, aby uzupełnić zapasy.
Uzupełnianie zapasów rozpoczyna się, gdy poziom jednego z paliw spadnie poniżej 60L oraz wszystkie samochody na stanowiskach zakończą tankowanie. Podczas uzupełnia zapasów blokowana jest możliwość rozpoczęcia tankowania.

Zakładam, że podczas uzupełniania zapasów:
- Samochody nie mogą rozpocząć pobierania paliwa.
- Samochody przy stanowiskach mają możliwość pójścia do kasy, jak i opuszczenia stacji.
- Samochody w kolejce mają możliwość zajęcia wolnego stanowiska.

3. Wykaz współdzielonych zasobów.

Zasobami współdzielonymi są:
- Semafory: „chron”, „stacja”, „stanowiska”, „kasy”, „tankowanie”, „uzupełnianie”, „zatankowane”.
- Zmienne w klasie „zasoby”.

4. Wykaz wyróżnionych sekcji krytycznych.

Sekcje krytyczne występują w klasach:
- Aplikacja – do dodania nowo utworzonego samochodu do kolejki.
- Samochod – do zajmowania miejsc, do pobrania paliwa, zmiany liczby tankujących samochodów.
- Uzupelnianie – do zmiany ilości paliwa na stacji

5. Wykaz obiektów synchronizacji.

Do synchronizacji używam:
-Semafory: „chron”, „stacja”, „stanowiska”, „kasy”, „tankowanie”, „uzupełnianie”, „zatankowane”.
-Słowa kluczowego „synchronized” w metodzie „stworz_samochod”.
-Oczekiwania aktywnego.

6. Wykaz procesów sekwencyjnych.

Procesami sekwencyjnymi w moim programie są:
- N obiektów klasy „Samochod”.

7. Listing programu.

```
package com.example.projekt;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;

public class main extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new
FXMLLoader(Aplikacja.class.getResource("aplikacja.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 800, 800);
        stage.setTitle("Stacja paliw");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch();
    }
}
```

```
package com.example.projekt;

import javafx.application.Platform;
import javafx.scene.control.Alert;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Rectangle;
import javafx.scene.shape.StrokeType;
import javafx.scene.text.Text;

import java.awt.*;
import java.util.ArrayList;
import java.util.Random;
import java.util.concurrent.Semaphore;

public class Aplikacja extends Thread {
    static int bokKwadratu = 75;
    Rectangle rec;
    Text text, text1;
    static ArrayList<Rectangle> samochodyList;
    static ArrayList<Rectangle> samochodyKolejkaList;
    static ArrayList<Text> textSamochodyKolejkaList;
    static ArrayList<Text> textSamochodyList;
    static ArrayList<Text> textKierowcaList;
    static ArrayList<Circle> kierowcaList;
    static Pane pane;
    static ArrayList<Text> textPaliwoSamochodyList;
    static ArrayList<Text> textPaliwoSamochodyKolejkaList;
    ArrayList<Integer> kolejka;
    Semaphore chron, stacja;
    int N, idPaliwa, litry;
    Uzupelnienie uzupelnienie;
    static ArrayList<Samochod> samochody;
```

```
    public Aplikacja(Pane pane) {
        samochodyKolejkaList = new ArrayList<>();
        samochodyList = new ArrayList<>();
        textSamochodyKolejkaList = new ArrayList<>();
        textSamochodyList = new ArrayList<>();
        kierowcaList=new ArrayList<>();
        textKierowcaList=new ArrayList<>();
        Aplikacja.pane=pane;
        kolejka = new ArrayList<>();
        textPaliwoSamochodyList=new ArrayList<>();
        textPaliwoSamochodyKolejkaList=new ArrayList<>();
        chron = new Semaphore(1);
        samochody=new ArrayList<>();
    }

    synchronized private void stworz_samochod(int id, int idPaliwa, int
litry){
        //samochody
        try {
            chron.acquire();
        } catch (InterruptedException e) {
            return;
        }
        Platform.runLater(()->{
            rec=new
Rectangle(ControllerMain.kolejkaList.get(samochodyKolejkaList.size()).getX(
)+5, ControllerMain.kolejkaList.get(samochodyList.size()).getY()+5,
bokKwadratu, bokKwadratu);
            Random random = new Random();
            switch (random.nextInt(7)) {
                case 0 -> rec.setFill(Color.RED);
                case 1 -> rec.setFill(Color.GREEN);
                case 2 -> rec.setFill(Color.YELLOW);
                case 3 -> rec.setFill(Color.WHITE);
                case 4 -> rec.setFill(Color.PINK);
                case 5 -> rec.setFill(Color.BLUE);
                case 6 -> rec.setFill(Color.LIGHTBLUE);
                case 7 -> rec.setFill(Color.LIGHTGREY);
            }
            rec.setStrokeType(StrokeType.INSIDE);
            rec.setStrokeWidth(1);
            rec.setStroke(Color.BLACK);
            rec.setAccessibleText(String.valueOf(id));
            samochodyKolejkaList.add(rec);
            samochodyList.add(rec);

            text=new Text(String.valueOf(id));
            text.setX(rec.getX()+2);
            text.setY(rec.getY()+text.getBoundsInLocal().getHeight()-3);
            text.setFill(Color.BLACK);

            text1=new
Text(ControllerUstawienia.zasoby.getRodzajePaliw()[idPaliwa]+":
"+litry+"L");
            text1.setX(rec.getX()+3);
            text1.setY(rec.getY()+bokKwadratu-3);
            text1.setFill(Color.BLACK);

            textSamochodyKolejkaList.add(text);
            textSamochodyList.add(text);
```

```
            textPaliwoSamochodyKolejkaList.add(text1);
            textPaliwoSamochodyList.add(text1);

            pane.getChildren().add(rec);
            pane.getChildren().add(text);
            pane.getChildren().add(text1);
            chron.release();
        });
    }

    private void poczatek() {
        int M=ControllerUstawienia.zasoby.getM(); //liczba stanowisk z
dystrybutorami
        N=ControllerUstawienia.zasoby.getN(); //maksymalna liczba
samochodów które mogą przebywać na stacji (liczba stanowisk + kolejka)
        int K=ControllerUstawienia.zasoby.getK(); //liczba kas
        int ileUzupelnia=ControllerUstawienia.zasoby.getIleUzupelnia();
//czas uzupełniania zasobów (blokowania dystrybutorów)
        int ileTankuje=ControllerUstawienia.zasoby.getIleTankuje(); //czas
potrzebny do zatankowania 1L paliwa
        int ilePlaci=ControllerUstawienia.zasoby.getIlePlaci(); //czas
potrzebny na zapłacenie

        Semaphore stanowiska = new Semaphore(M); // liczba stanowisk
        Semaphore kasy = new Semaphore(K); // liczba kas
        stacja = new Semaphore(N); // liczba samochodów na stacji
        Semaphore tankowanie = new Semaphore(0); // powiadamia obiekt
samochod, że może zacząć tankowanie
        Semaphore uzupelnianie = new Semaphore(0); // sprawdza, czy
potrzebne jest uzupełnienie zapasów paliwa na stacji
        Semaphore zatankowane = new Semaphore(0); // powiadamia wątek
"uzupelnianie", że zaktualizowano ilość paliwa

        boolean[] wolneStanowisko=new boolean[M];
        boolean[] wolnaKasa=new boolean[K];
        for (int j=0; j<M; j++)
            wolneStanowisko[j]=true;
        for (int j=0; j<K; j++)
            wolnaKasa[j]=true;

        Samochod samochod;
        uzupelnienie = new Uzupelnienie(ileUzupelnia, chron, tankowanie,
uzupelnianie, zatankowane);
        uzupelnienie.start();
        int i=0;
        boolean loop=true;
        while(loop){
            try {
                stacja.acquire();
                Random random = new Random();

idPaliwa=random.nextInt(ControllerUstawienia.zasoby.getRodzajePaliw().lengt
h);
                litry = random.nextInt(60)+1; //<1, 60>
                chron.acquire();
                kolejka.add(i);
                chron.release();
                stworz_samochod(i, idPaliwa, litry);
                samochod = new Samochod(i, stanowiska, kasy, stacja,
ileTankuje, ilePlaci, ileUzupelnia, chron,
                        tankowanie, uzupelnianie, zatankowane, kolejka,
```

```
wolneStanowisko, pane, wolnaKasa, idPaliwa, litry);
                samochody.add(samochod);
                samochod.start();
                if(ControllerUstawienia.zasoby.getAuta()>0 &&
i==ControllerUstawienia.zasoby.getAuta()-1)
                    loop=false;
                i++;
            } catch (InterruptedException e) {
                uzupelnienie.interrupt();
                for (Samochod x : samochody){
                    x.interrupt();
                }
                Thread.currentThread().stop();
            }
        }
    }

    public void koniec(){
        while (stacja.availablePermits()!=N);
        uzupelnienie.interrupt();
        Platform.runLater(()->{
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setTitle("Koniec Symulacji");
            alert.setHeaderText(null);
            alert.setContentText("Symulacja zakończona!");
            alert.show();

            final Runnable runnable = (Runnable)
Toolkit.getDefaultToolkit().getDesktopProperty("win.sound.default");
            runnable.run();
            ControllerMain.btnStartS.setDisable(false);
            ControllerMain.btnStopS.setDisable(true);
            ControllerMain.btnSettingS.setDisable(false);
            ControllerMain.flagConfig=false;
        });
    }

    @Override
    public void run(){
        poczatek();
        koniec();
    }
}
```

```
package com.example.projekt;

import javafx.animation.ParallelTransition;
import javafx.animation.TranslateTransition;
import javafx.application.Platform;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Rectangle;
import javafx.scene.shape.StrokeType;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.util.Duration;

import java.util.ArrayList;
import java.util.Objects;
```

```
import java.util.Random;
import java.util.concurrent.Semaphore;

public class Samochod extends Thread{
    int id, promienKierowcy, ileTankuje, ilePlaci, ileUzupelnia, idPaliwa,
litry, stanowisko, kasa;
    Semaphore stanowiska, kasy, stacja, tankowanie, chron, uzupelnianie,
zatankowane;
    String paliwo;
    boolean[] wolneStanowisko;
    boolean[] wolnaKasa;
    ArrayList<Integer> kolejka;
    Pane pane;
    Circle circle;
    Text text;

    public Samochod(int id, Semaphore stanowiska, Semaphore kasy, Semaphore
stacja, int ileTankuje, int ilePlaci,
                    int ileUzupelnia, Semaphore chron, Semaphore
tankowanie, Semaphore uzupelnianie, Semaphore zatankowane,
ArrayList<Integer> kolejka,
                    boolean[] wolneStanowisko, Pane pane, boolean[]
wolnaKasa, int idPaliwa, int litry) {
        this.id = id;
        this.stanowiska = stanowiska;
        this.kasy = kasy;
        this.stacja = stacja;
        this.ileTankuje = ileTankuje;
        this.ilePlaci = ilePlaci;
        this.ileUzupelnia=ileUzupelnia;
        this.idPaliwa=idPaliwa;
        this.paliwo = ControllerUstawienia.zasoby.getPaliwo(idPaliwa);
        this.litry=litry;
        this.chron=chron;
        this.kolejka=kolejka;
        this.wolneStanowisko=wolneStanowisko;
        this.wolnaKasa=wolnaKasa;
        this.pane=pane;
        promienKierowcy=20;
        this.tankowanie=tankowanie;
        this.uzupelnianie = uzupelnianie;
        this.zatankowane = zatankowane;
    }

    private void zajecieStanowiska(int st){
        try {
            chron.acquire();
        } catch (InterruptedException e) {
            return;
        }
        int h=-1;
        for(int i=0; i<Aplikacja.samochodyKolejkaList.size(); i++){

if(Objects.equals(Aplikacja.samochodyKolejkaList.get(i).getAccessibleText()
, String.valueOf(id))){
                h=i;
                break;
            }
        }
        Rectangle samochodKolejka=Aplikacja.samochodyKolejkaList.get(h);
        Text textSamochodKolejka=Aplikacja.textSamochodyKolejkaList.get(h);
```

```
        Text
textPaliwoSamochodKolejka=Aplikacja.textPaliwoSamochodyKolejkaList.get(h);
        Aplikacja.samochodyKolejkaList.remove(h);
        Aplikacja.textSamochodyKolejkaList.remove(h);
        Aplikacja.textPaliwoSamochodyKolejkaList.remove(h);
        kolejka.remove(h);

        Platform.runLater(()->{
            TranslateTransition translate = new
TranslateTransition(Duration.millis(ControllerUstawienia.zasoby.getIleAnim(
)));

translate.setToX(ControllerMain.stanowiskaList.get(st).getX()+5-
samochodKolejka.getX());

translate.setToY(ControllerMain.stanowiskaList.get(st).getY()+5-
samochodKolejka.getY());
            TranslateTransition translateText = new
TranslateTransition(Duration.millis(ControllerUstawienia.zasoby.getIleAnim(
)));

translateText.setToX(ControllerMain.stanowiskaList.get(st).getX()+5-
samochodKolejka.getX());

translateText.setToY(ControllerMain.stanowiskaList.get(st).getY()+5-
samochodKolejka.getY());
            TranslateTransition translateTextPaliwo = new
TranslateTransition(Duration.millis(ControllerUstawienia.zasoby.getIleAnim(
)));

translateTextPaliwo.setToX(ControllerMain.stanowiskaList.get(st).getX()+5-
samochodKolejka.getX());

translateTextPaliwo.setToY(ControllerMain.stanowiskaList.get(st).getY()+5-
samochodKolejka.getY());
            ParallelTransition transition = new
ParallelTransition(samochodKolejka,translate);
            ParallelTransition transitionText = new
ParallelTransition(textSamochodKolejka,translateText);
            ParallelTransition transitionTextPaliwo = new
ParallelTransition(textPaliwoSamochodKolejka,translateTextPaliwo);
            transition.play();
            transitionText.play();
            transitionTextPaliwo.play();
            transition.setOnFinished(actionEvent -> {
                stworz_Kierowce(id, stanowisko);
                moveKolejka();
            });
        });
    }

    private void moveKolejka(){
        Platform.runLater(()->{
            for(int i=0; i<Aplikacja.samochodyKolejkaList.size(); i++){

Aplikacja.samochodyKolejkaList.get(i).setX(ControllerMain.kolejkaList.get(i
).getX()+5);

Aplikacja.textSamochodyKolejkaList.get(i).setX(ControllerMain.kolejkaList.g
et(i).getX()+5+3);
```

```
Aplikacja.textPaliwoSamochodyKolejkaList.get(i).setX(ControllerMain.kolejka
List.get(i).getX()+5+3);
            }
            chron.release();
        });
    }

    private void stworz_Kierowce(int id, int stanowisko){
        Platform.runLater(()->{
            circle=new
Circle(ControllerMain.stanowiskaList.get(stanowisko).getX()+5+(float)Contro
llerMain.bokKwadratu/2,ControllerMain.stanowiskaList.get(stanowisko).getY()
+5+(float)ControllerMain.bokKwadratu/2,promienKierowcy);
            Random random = new Random();
            switch (random.nextInt(4)) {
                case 0 -> circle.setFill(Color.WHITE);
                case 1 -> circle.setFill(Color.BROWN);
                case 2 -> circle.setFill(Color.YELLOW);
                case 3 -> circle.setFill(Color.LIGHTPINK);
            }
            circle.setStrokeType(StrokeType.INSIDE);
            circle.setStrokeWidth(1);
            circle.setStroke(Color.BLACK);
            circle.setAccessibleText(String.valueOf(id));
            Aplikacja.kierowcaList.add(circle);
            text=new Text(String.valueOf(id));
            text.setX(circle.getCenterX()-
text.getBoundsInLocal().getWidth()/2-2);

text.setY(circle.getCenterY()+text.getBoundsInLocal().getHeight()/2-3);
            text.setFont(new Font(15));
            text.setFill(Color.BLACK);
            Aplikacja.textKierowcaList.add(text);
            pane.getChildren().add(circle);
            pane.getChildren().add(text);
        });
    }

    private void zajecieKasy(int id, int kasa){
        try {
            chron.acquire();
        } catch (InterruptedException e) {
            return;
        }
        int h=-1;
        for (int i=0; i<Aplikacja.kierowcaList.size(); i++){

if(Objects.equals(Aplikacja.kierowcaList.get(i).getAccessibleText(),
String.valueOf(id))){
                h=i;
                break;
            }
        }
        Circle kierowca=Aplikacja.kierowcaList.get(h);
        Text textKierowca=Aplikacja.textKierowcaList.get(h);
        chron.release();
        Platform.runLater(()->{
            TranslateTransition translate = new
TranslateTransition(Duration.millis(ControllerUstawienia.zasoby.getIleAnim(
)));
            translate.setToX(ControllerMain.kasyList.get(kasa).getX()-
```

```
textKierowca.getX()+(float)ControllerMain.szerKasy/2-
textKierowca.getBoundsInLocal().getWidth()/2);
            translate.setToY(ControllerMain.kasyList.get(kasa).getY()-
textKierowca.getY()+(float)ControllerMain.wysKasy/2+textKierowca.getBoundsI
nLocal().getHeight()/4);
            TranslateTransition translateText = new
TranslateTransition(Duration.millis(ControllerUstawienia.zasoby.getIleAnim(
)));
            translateText.setToX(ControllerMain.kasyList.get(kasa).getX()-
textKierowca.getX()+(float)ControllerMain.szerKasy/2-
textKierowca.getBoundsInLocal().getWidth()/2);
            translateText.setToY(ControllerMain.kasyList.get(kasa).getY()-
textKierowca.getY()+(float)ControllerMain.wysKasy/2+textKierowca.getBoundsI
nLocal().getHeight()/4);
            ParallelTransition transition = new
ParallelTransition(kierowca,translate);
            ParallelTransition transitionText = new
ParallelTransition(textKierowca,translateText);
            transition.play();
            transitionText.play();
        });
    }

    private void wyjazd(int id, int kasa, int stanowisko){
        try {
            chron.acquire();
        } catch (InterruptedException e) {
            return;
        }
        int h=-1;
        for(int i=0; i<Aplikacja.kierowcaList.size(); i++){

if(Objects.equals(Aplikacja.kierowcaList.get(i).getAccessibleText(),
String.valueOf(id))){
                h=i;
                break;
            }
        }
        int hSamochod=-1;
        for (int i=0; i<Aplikacja.samochodyList.size(); i++){

if(Objects.equals(Aplikacja.samochodyList.get(i).getAccessibleText(),
String.valueOf(id))){
                hSamochod=i;
                break;
            }
        }
        Circle kierowca = Aplikacja.kierowcaList.get(h);
        Text textKierowca = Aplikacja.textKierowcaList.get(h);
        Rectangle samochod = Aplikacja.samochodyList.get(hSamochod);
        Text textSamochodu = Aplikacja.textSamochodyList.get(hSamochod);
        Text textPaliwoSamochodu =
Aplikacja.textPaliwoSamochodyList.get(hSamochod);
        Aplikacja.kierowcaList.remove(h);
        Aplikacja.textKierowcaList.remove(h);
        Aplikacja.samochodyList.remove(hSamochod);
        Aplikacja.textSamochodyList.remove(hSamochod);
        Aplikacja.textPaliwoSamochodyList.remove(hSamochod);
        chron.release();

        Platform.runLater(()->{
```

```
            TranslateTransition translate = new
TranslateTransition(Duration.millis((float)ControllerUstawienia.zasoby.getI
leAnim()/2));
            TranslateTransition translateText = new
TranslateTransition(Duration.millis((float)ControllerUstawienia.zasoby.getI
leAnim()/2));
            translate.setToX(0);
            translate.setToY(0);
            translateText.setToX(0);
            translateText.setToY(0);
            ParallelTransition transition = new
ParallelTransition(kierowca,translate);
            ParallelTransition transitionText = new
ParallelTransition(textKierowca,translateText);
            wolnaKasa[kasa]=true;
            kasy.release();
            transition.play();
            transitionText.play();
            transition.setOnFinished(actionEvent -> {
                translate.setToX(0);
                translate.setToY(pane.getHeight() / 2 + promienKierowcy);
                ParallelTransition transition1 = new
ParallelTransition(kierowca, translate);
                TranslateTransition translateSamochod = new
TranslateTransition(Duration.millis((double)
ControllerUstawienia.zasoby.getIleAnim() / 2));

translateSamochod.setToX(ControllerMain.stanowiskaList.get(stanowisko).getX
() - samochod.getX() + 5);
                translateSamochod.setToY(pane.getHeight() - samochod.getY()
+ (float)ControllerMain.bokKwadratu / 2);
                ParallelTransition transitionSamochod = new
ParallelTransition(samochod, translateSamochod);
                transitionSamochod.play();
                transition1.play();
                transition1.setOnFinished(actionEvent2 -> {
                    wolneStanowisko[stanowisko] = true;
                    stanowiska.release();
                    stacja.release();
                });
            });
            transitionText.setOnFinished(actionEvent -> {
                translateText.setToX(0);
                translateText.setToY(pane.getHeight()/2+promienKierowcy);
                ParallelTransition transitionText2 = new
ParallelTransition(textKierowca,translateText);
                TranslateTransition translateTextSamochod = new
TranslateTransition(Duration.millis((float)ControllerUstawienia.zasoby.getI
leAnim()/2));

translateTextSamochod.setToX(ControllerMain.stanowiskaList.get(stanowisko).
getX()-samochod.getX()+5);
                translateTextSamochod.setToY(pane.getHeight()-
samochod.getY()+(float)ControllerMain.bokKwadratu/2);
                ParallelTransition transitionTextSamochod = new
ParallelTransition(textSamochodu,translateTextSamochod);
                TranslateTransition translateTextPaliwo=new
TranslateTransition(Duration.millis((float)ControllerUstawienia.zasoby.getI
leAnim()/2));

translateTextPaliwo.setToX(ControllerMain.stanowiskaList.get(stanowisko).ge
```

```
tX()-samochod.getX()+5);
                translateTextPaliwo.setToY(pane.getHeight()-
samochod.getY()+(float)ControllerMain.bokKwadratu/2);
                ParallelTransition transitionTextPaliwo = new
ParallelTransition(textPaliwoSamochodu,translateTextPaliwo);
                transitionTextPaliwo.play();
                transitionTextSamochod.play();
                transitionText2.play();
            });
        });
    }

    @Override
    public void run() {
        try {
            stanowiska.acquire();
            for(int i=0; i<wolneStanowisko.length; i++){
                if(wolneStanowisko[i]){
                    wolneStanowisko[i]=false;
                    stanowisko=i;
                    break;
                }
            }
            zajecieStanowiska(stanowisko);
            uzupelnianie.release();
            tankowanie.acquire();
            chron.acquire();

ControllerUstawienia.zasoby.setTankujaceSamochody(ControllerUstawienia.zaso
by.getTankujaceSamochody()+1);

ControllerUstawienia.zasoby.setPozostalePaliwo(ControllerUstawienia.zasoby.
getPozostalePaliwo(idPaliwa) - litry, idPaliwa);

ControllerMain.textPozostalePaliwoList[idPaliwa].setText(String.valueOf(Con
trollerUstawienia.zasoby.getPozostalePaliwo(idPaliwa)));
            chron.release();
            zatankowane.release();
            Thread.sleep((long) ileTankuje * litry +
ControllerUstawienia.zasoby.getIleAnim());
            chron.acquire();

ControllerUstawienia.zasoby.setTankujaceSamochody(ControllerUstawienia.zaso
by.getTankujaceSamochody()-1);
            chron.release();

            kasy.acquire();
            for (int i=0; i<wolnaKasa.length; i++){
                if(wolnaKasa[i]){
                    wolnaKasa[i]=false;
                    kasa=i;
                    break;
                }
            }
            zajecieKasy(id,kasa);
            Thread.sleep(ilePlaci +
ControllerUstawienia.zasoby.getIleAnim());
            wyjazd(id,kasa,stanowisko);
            Aplikacja.samochody.remove(Samochod.this);
        } catch (InterruptedException e) {
            //e.printStackTrace();
```

```
            stacja.release();
            Thread.currentThread().stop();
        }
    }
}
```

```
package com.example.projekt;

import java.util.concurrent.Semaphore;

public class Uzupelnienie extends Thread{
    Semaphore tankowanie, chron, uzupelnianie, zatankowane;
    int ileUzupelnia;

    public Uzupelnienie(int ileUzupelnia, Semaphore chron, Semaphore
tankowanie, Semaphore uzupelnianie, Semaphore zatankowane) {
        this.ileUzupelnia = ileUzupelnia;
        this.chron = chron;
        this.tankowanie = tankowanie;
        this.uzupelnianie = uzupelnianie;
        this.zatankowane = zatankowane;
    }

    @Override
    public void run() {
        while(true) {
            try {
                uzupelnianie.acquire();
                if(ControllerUstawienia.zasoby.koniecPaliwa()) {

while(ControllerUstawienia.zasoby.getTankujaceSamochody()!=0) {
                        if(this.isInterrupted())
                            Thread.currentThread().stop();
                    }
                    Thread.sleep(ileUzupelnia);
                    chron.acquire();
                    for (int j = 0; j <
ControllerUstawienia.zasoby.getRodzajePaliw().length; j++) {

ControllerUstawienia.zasoby.setPozostalePaliwo(ControllerUstawienia.zasoby.
getMaxPaliwa(), j);

ControllerMain.textPozostalePaliwoList[j].setText(String.valueOf(Controller
Ustawienia.zasoby.getPozostalePaliwo(j)));
                    }
                    chron.release();
                }
                tankowanie.release();
                zatankowane.acquire();
            } catch (InterruptedException e) {
                //e.printStackTrace();
                Thread.currentThread().stop();
            }
        }
    }
}
```

```
package com.example.projekt;
```

```java
import java.util.Arrays;

public class Zasoby {
    private int M; //liczba stanowisk z dystrybutorami
    private int N; //maksymalna liczba samochodów które mogą przebywać na stacji (liczba stanowisk + kolejka)
    private int K; //liczba kas
    private int ileUzupelnia; //czas uzupełniania zasobów (blokowania dystrybutorów)
    private int ileTankuje; //czas potrzebny do zatankowania 1L paliwa
    private int ilePlaci; //czas potrzebny na zapłacenie
    private int maxPaliwa; //maksymalna ilość paliwa w zasobach
    private int ileAnim;
    private int TankujaceSamochody, Auta;
    private String[] rodzajePaliw; //rodzaje paliwa
    private int[] pozostalePaliwo;

    public Zasoby() {
        this.TankujaceSamochody=0;
        this.Auta = 0;
        this.M = 4;
        this.N = 8;
        this.K = 2;
        this.ileUzupelnia = 6000;
        this.ileTankuje = 100;
        this.ilePlaci = 3000;
        this.maxPaliwa = 200;
        this.ileAnim = 600;
        this.rodzajePaliw = new String[]{"95", "ON", "LPG"};
    }

    boolean koniecPaliwa(){
        for(int j=0; j<rodzajePaliw.length; j++){
            if(pozostalePaliwo[j]<60) return true;
        }
        return false;
    }

    public String[] getRodzajePaliw() {
        return rodzajePaliw;
    }

    public String getPaliwo(int i){
        return rodzajePaliw[i];
    }

    public int getTankujaceSamochody() {return this.TankujaceSamochody;}

    public void setTankujaceSamochody(int TankujaceSamochody) {this.TankujaceSamochody = TankujaceSamochody;}

    public int getPozostalePaliwo(int i) {
        return pozostalePaliwo[i];
    }

    public void setPozostalePaliwo(int pozostalePaliwo, int i) {
        this.pozostalePaliwo[i] = pozostalePaliwo;
    }

    public int getIleUzupelnia() {
        return ileUzupelnia;
```

```
    }

    public void setIleUzupelnia(int ileUzupelnia) {
        this.ileUzupelnia = ileUzupelnia;
    }

    public int getIleTankuje() {
        return ileTankuje;
    }

    public void setIleTankuje(int ileTankuje) {
        this.ileTankuje = ileTankuje;
    }

    public int getIlePlaci() {
        return ilePlaci;
    }

    public void setIlePlaci(int ilePlaci) {
        this.ilePlaci = ilePlaci;
    }

    public int getMaxPaliwa() {
        return maxPaliwa;
    }

    public void setMaxPaliwa(int maxPaliwa) {
        this.maxPaliwa = maxPaliwa;
    }

    public int getAuta() {
        return Auta;
    }

    public void setAuta(int Auta) {
        this.Auta = Auta;
    }

    public int getIleAnim() {
        return ileAnim;
    }

    public void setIleAnim(int ileAnim) {
        this.ileAnim = ileAnim;
    }

    public void setRodzajePaliw(String[] rodzajePaliw) {
        this.rodzajePaliw = rodzajePaliw;

        this.pozostalePaliwo=new int[rodzajePaliw.length];
        Arrays.fill(this.pozostalePaliwo, this.maxPaliwa);
    }

    public int getM() {
        return M;
    }

    public void setM(int m) {
        M = m;
    }
```

```
    public int getK() {
        return K;
    }

    public void setK(int k) {
        K = k;
    }

    public int getN() {
        return N;
    }

    public void setN(int n) {
        N = n;
    }
}
```

```
package com.example.projekt;

import javafx.event.Event;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.shape.StrokeType;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javafx.stage.WindowEvent;

import java.awt.*;
import java.io.IOException;
import java.util.ArrayList;

public class ControllerMain {
    public Button btnStart;
    public Button btnStop;
    public Button btnSettings;
    public Pane pane;
    Zasoby zasoby;
    static ArrayList<Rectangle> kolejkaList;
    static ArrayList<Rectangle> stanowiskaList;
    static ArrayList<Rectangle> kasyList;
    ArrayList<Text> textkolejkaList;
    static Text[] textPozostalePaliwoList;

    Aplikacja aplikacja;

    static int bokKwadratu;
    static int wysKasy;
    static int szerKasy;

    static boolean flagConfig=false;

    static Button btnStartS;
```

```
    static Button btnStopS;
    static Button btnSettingsS;

    public void initialize(){
        btnStartS=btnStart;
        btnStopS=btnStop;
        btnSettingsS=btnSettings;
    }

    public void startSymulation(Event actionEvent) throws IOException {
        if (flagConfig) {
            btnStart.setDisable(true);
            btnStop.setDisable(false);
            btnSettings.setDisable(true);
            aplikacja = new Aplikacja(pane);
            aplikacja.start();
        } else zasoby(actionEvent);
    }

    public void stopSymulation(Event actionEvent) {
        btnStart.setDisable(false);
        btnStop.setDisable(true);
        btnSettings.setDisable(false);
        flagConfig=false;
        aplikacja.interrupt();

        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Koniec Symulacji");
        alert.setHeaderText(null);
        alert.setContentText("Symulacja zakończona!");
        alert.show();

        final Runnable runnable = (Runnable)
Toolkit.getDefaultToolkit().getDesktopProperty("win.sound.default");
        runnable.run();
    }

    public void zasoby(Event actionEvent) throws IOException {
        FXMLLoader fxmlLoader = new
FXMLLoader(main.class.getResource("ustawienia.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 400, 510);
        Stage stage = new Stage();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setResizable(false);
        stage.setTitle("Ustawienia");
        stage.setScene(scene);

        stage.setOnHidden(new EventHandler<WindowEvent>() {
            public void handle(WindowEvent we) {
                pane.getChildren().clear();
                zasoby=ControllerUstawienia.zasoby;
                bokKwadratu = 75;
                int odstep = 10;
                Rectangle rec;

                //kolejka
                kolejkaList = new ArrayList<>();
                textkolejkaList = new ArrayList<>();
                double startingPointX = pane.getWidth() / 2 - zasoby.getN()
* (bokKwadratu+10) / 2 - (zasoby.getN() - 1) * odstep / 2;
                double startingPointY = 20;
```

```
                for (int i = 0; i < zasoby.getN(); i++) {
                    rec = new Rectangle(startingPointX + i * odstep + i *
(bokKwadratu+10), startingPointY, (bokKwadratu+10), (bokKwadratu+10));
                    kolejkaList.add(rec);
                }
                for (Rectangle x : kolejkaList) {
                    x.setFill(Color.LIGHTBLUE);
                    x.setStrokeType(StrokeType.INSIDE);
                    x.setStrokeWidth(1);
                    x.setStroke(Color.BLACK);
                    pane.getChildren().add(x);
                }

                //stanowiska
                stanowiskaList = new ArrayList<>();
                startingPointX=pane.getWidth()/2-
(float)zasoby.getM()*(bokKwadratu+10)/2-(float)(zasoby.getM()-1)*odstep/2;
                startingPointY=pane.getHeight()/2-
(float)(bokKwadratu+10)/2;
                for(int i=0; i<zasoby.getM(); i++){
                    rec=new
Rectangle(startingPointX+i*odstep+i*(bokKwadratu+10), startingPointY,
(bokKwadratu+10), (bokKwadratu+10));
                    stanowiskaList.add(rec);
                }
                for(Rectangle x : stanowiskaList){
                    x.setFill(Color.WHITE);
                    x.setStrokeType(StrokeType.INSIDE);
                    x.setStrokeWidth(1);
                    x.setStroke(Color.BLACK);

                    pane.getChildren().add(x);
                }

                //kasy
                wysKasy=75;
                szerKasy=50;
                kasyList = new ArrayList<>();
                startingPointX=pane.getWidth()-szerKasy;
                startingPointY=pane.getHeight()/2-zasoby.getK()*wysKasy/2-
(zasoby.getK()-1)*odstep/2;
                for(int i=0; i<zasoby.getK(); i++){
                    rec=new Rectangle(startingPointX,
startingPointY+i*odstep+i*wysKasy, szerKasy, wysKasy);
                    kasyList.add(rec);
                }
                for(Rectangle x : kasyList){
                    x.setFill(Color.ORANGERED);
                    x.setStrokeType(StrokeType.INSIDE);
                    x.setStrokeWidth(1);
                    x.setStroke(Color.BLACK);
                    pane.getChildren().add(x);
                }

                //pozostale paliwa
                Text text1;
                int max=0;
                for(String x : zasoby.getRodzajePaliw()) {
                    if (max < x.length()) max = x.length();
                }
                Text text;
```

```
                textPozostalePaliwoList=new
Text[zasoby.getRodzajePaliw().length];
                for(int i=0; i<zasoby.getRodzajePaliw().length; i++){
                    text=new Text(zasoby.getRodzajePaliw()[i]+":");
                    text.setFont(Font.font(16));
                    text.setX(3);
                    text.setY(pane.getHeight()-10-
i*text.getBoundsInLocal().getHeight());

                    text1=new
Text(String.valueOf(zasoby.getPozostalePaliwo(i)));
                    text1.setFont(Font.font(16));
                    text1.setX(max*16);
                    text1.setY(text.getY());

text1.setAccessibleText(text.getText().replace(":",""));

                    pane.getChildren().add(text);
                    pane.getChildren().add(text1);
                    textPozostalePaliwoList[i]=text1;
                }
                flagConfig=true;
            }
        });
        stage.show();
    }
}
```

```
package com.example.projekt;

import javafx.event.ActionEvent;
import javafx.scene.Node;
import javafx.scene.control.TextField;
import javafx.scene.input.KeyEvent;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import java.io.*;
import java.util.Arrays;

public class ControllerUstawienia {
    public TextField tfM, tfK, tfN, tfIleUzupelnia, tfIleTankuje,
tfIlePlaci, tfMaxPaliwa, tfRodzajePaliw, tfIleAnim, tfAuta;
    public static Zasoby zasoby = new Zasoby();
    public void initialize(){
        initializeSettings();
    }

    private void initializeSettings(){
        tfAuta.setText(String.valueOf(zasoby.getAuta()));
```

```
        tfM.setText(String.valueOf(zasoby.getM()));
        tfK.setText(String.valueOf(zasoby.getK()));
        tfN.setText(String.valueOf(zasoby.getN()));
        tfIleUzupelnia.setText(String.valueOf(zasoby.getIleUzupelnia()));
        tfIleTankuje.setText(String.valueOf(zasoby.getIleTankuje()));
        tfIlePlaci.setText(String.valueOf(zasoby.getIlePlaci()));
        tfMaxPaliwa.setText(String.valueOf(zasoby.getMaxPaliwa()));
        tfIleAnim.setText(String.valueOf(zasoby.getIleAnim()));

tfRodzajePaliw.setText(Arrays.toString(zasoby.getRodzajePaliw()).replaceAll
("[\\[\\]]",""));
    }

    private void saveSetting(){
        zasoby.setAuta(Integer.parseInt(tfAuta.getText()));
        zasoby.setM(Integer.parseInt(tfM.getText()));
        zasoby.setK(Integer.parseInt(tfK.getText()));
        zasoby.setN(Integer.parseInt(tfN.getText()));
        zasoby.setIleUzupelnia(Integer.parseInt(tfIleUzupelnia.getText()));
        zasoby.setIleTankuje(Integer.parseInt(tfIleTankuje.getText()));
        zasoby.setIlePlaci(Integer.parseInt(tfIlePlaci.getText()));
        zasoby.setMaxPaliwa(Integer.parseInt(tfMaxPaliwa.getText()));
        zasoby.setIleAnim(Integer.parseInt(tfIleAnim.getText()));
        zasoby.setRodzajePaliw(tfRodzajePaliw.getText().replaceAll("
","").split(","));
    }

    public void setSettings(ActionEvent actionEvent) {
        Node node = (Node) actionEvent.getSource();
        Stage thisStage = (Stage) node.getScene().getWindow();
        saveSetting();
        thisStage.close();
    }

    public static void writeXml(Document document, OutputStream
outputStream) throws TransformerException {
        TransformerFactory transformerFactory =
TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION,
"yes");
        DOMSource source = new DOMSource(document);
        StreamResult result = new StreamResult(outputStream);
        transformer.transform(source, result);
    }

    public void saveToFileSettings(ActionEvent actionEvent) throws
IOException, ParserConfigurationException, TransformerException {
        saveSetting();

        Node node = (Node) actionEvent.getSource();
        Stage thisStage = (Stage) node.getScene().getWindow();

        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Zapisz ustawienia");
        fileChooser.getExtensionFilters().addAll(new
FileChooser.ExtensionFilter("Ustawienia", "*.xml"));
        File file = fileChooser.showSaveDialog(thisStage);
        if(file==null) return;
```

```
        System.out.println(file.getAbsolutePath());
        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();

        Document document = documentBuilder.newDocument();
        Element rootElement = document.createElement("stacja");
        document.appendChild(rootElement);
        Element Auta = document.createElement("Auta");
        Auta.setTextContent(String.valueOf(zasoby.getAuta()));
        rootElement.appendChild(Auta);
        Element M = document.createElement("M");
        M.setTextContent(String.valueOf(zasoby.getM()));
        rootElement.appendChild(M);
        Element K = document.createElement("K");
        K.setTextContent(String.valueOf(zasoby.getK()));
        rootElement.appendChild(K);
        Element N = document.createElement("N");
        N.setTextContent(String.valueOf(zasoby.getN()));
        rootElement.appendChild(N);
        Element IleUzupelnia = document.createElement("IleUzupelnia");

IleUzupelnia.setTextContent(String.valueOf(zasoby.getIleUzupelnia()));
        rootElement.appendChild(IleUzupelnia);
        Element IleTankuje = document.createElement("IleTankuje");
        IleTankuje.setTextContent(String.valueOf(zasoby.getIleTankuje()));
        rootElement.appendChild(IleTankuje);
        Element IlePlaci = document.createElement("IlePlaci");
        IlePlaci.setTextContent(String.valueOf(zasoby.getIlePlaci()));
        rootElement.appendChild(IlePlaci);
        Element MaxPaliwa = document.createElement("MaxPaliwa");
        MaxPaliwa.setTextContent(String.valueOf(zasoby.getMaxPaliwa()));
        rootElement.appendChild(MaxPaliwa);
        Element IleAnim = document.createElement("ileAnim");
        IleAnim.setTextContent(String.valueOf(zasoby.getIleAnim()));
        rootElement.appendChild(IleAnim);
        Element RodzajePaliw = document.createElement("RodzajePaliw");

RodzajePaliw.setTextContent(Arrays.toString(zasoby.getRodzajePaliw()));
        rootElement.appendChild(RodzajePaliw);

        FileOutputStream outputStream = new FileOutputStream(file);
        writeXml(document, outputStream);

        thisStage.close();
    }

    public void loadFromFileSettings(ActionEvent actionEvent) throws
ParserConfigurationException, IOException, SAXException {
        Node node = (Node) actionEvent.getSource();
        Stage thisStage = (Stage) node.getScene().getWindow();

        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Wczytaj ustawienia");
        fileChooser.getExtensionFilters().addAll(new
FileChooser.ExtensionFilter("Ustawienia", "*.xml"));
        File file = fileChooser.showOpenDialog(thisStage);

        if(file==null) return;
```

```
        DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(file);


zasoby.setAuta(Integer.parseInt(document.getElementsByTagName("Auta").item(
0).getTextContent()));

zasoby.setM(Integer.parseInt(document.getElementsByTagName("M").item(0).get
TextContent()));

zasoby.setK(Integer.parseInt(document.getElementsByTagName("K").item(0).get
TextContent()));

zasoby.setN(Integer.parseInt(document.getElementsByTagName("N").item(0).get
TextContent()));

zasoby.setIleUzupelnia(Integer.parseInt(document.getElementsByTagName("IleU
zupelnia").item(0).getTextContent()));

zasoby.setIleTankuje(Integer.parseInt(document.getElementsByTagName("IleTan
kuje").item(0).getTextContent()));

zasoby.setIlePlaci(Integer.parseInt(document.getElementsByTagName("IlePlaci
").item(0).getTextContent()));

zasoby.setMaxPaliwa(Integer.parseInt(document.getElementsByTagName("MaxPali
wa").item(0).getTextContent()));

zasoby.setIleAnim(Integer.parseInt(document.getElementsByTagName("ileAnim")
.item(0).getTextContent()));

zasoby.setRodzajePaliw(document.getElementsByTagName("RodzajePaliw").item(0
).getTextContent().replaceAll("[\\[\\]]","").replaceAll("
","").split(","));
        initializeSettings();

        thisStage.close();
    }

    public void validate(KeyEvent keyEvent) {
        char input=keyEvent.getCharacter().charAt(0);
        Object source = keyEvent.getSource();
        if(Character.isLetter(input)){
            String Oldtext = ((TextField)source).getText();
            String NewText = Oldtext.replaceAll("[^\\d]", "");
            tfM.setText(NewText);
            tfM.positionCaret(NewText.length());
        }
    }
}
```