**WOJSKOWA AKADEMIA TECHNICZNA**

**im. Jarosława Dąbrowskiego**

WYDZIAŁ CYBERNETYKI



**Laboratorium**

**Przedmiot:** Programowanie Współbieżne

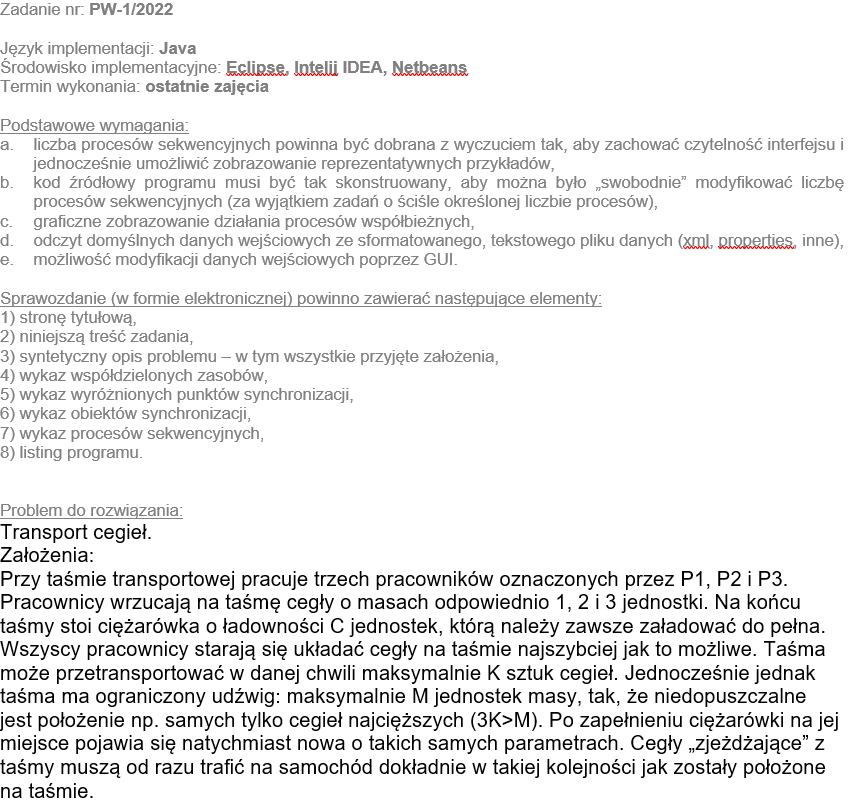
**Wykonał:** Filip Stańczak

**Grupa:** WCY19IJ3S1

**Data wykonania:** 16.01.2023r.

**Prowadzący:** dr inż. Jarosław Rulka

**Treść zadania:**



**Przyjęte założenia:**

Istnieje taśma transportowa na której trzech pracowników układa cegły. Na końcu taśmy znajduje się ciężarówka. Każdy pracownik układa cegły jak najszybciej, czas potrzebny na przeniesienie cegły jest zależny od jej masy. Taśma ma ograniczoną nośność a także maksymalną ilość cegieł znajdujących się na niej w danym momencie. Z taśmy cegły lądują do ciężarówki która ma określoną pojemność. W momencie jej całkowitego zapełnienia podstawiona jest nowa ciężarówka a poprzednia odjeżdża z towarem. Jeśli masa cegieł znajdujących się na taśmie wypełni ciężarówkę to pracownicy muszą poczekać aż dana ciężarówka się zapełni i podjedzie nowa. Ciężarówka nie może odebrać cegły w momencie dokładania nowej na taśmę. Pracownik nie może położyć cegły na taśmie gdy ciężarówka zabiera cegłę z taśmy.

**Wykaz współdzielonych zasobów:**

Taśma transportowa:

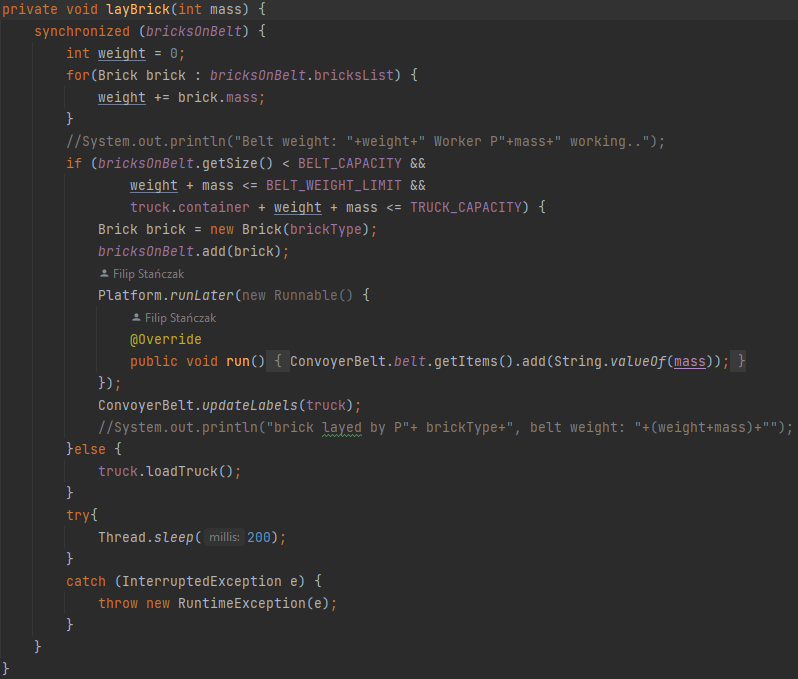
****

Ciężarówka:

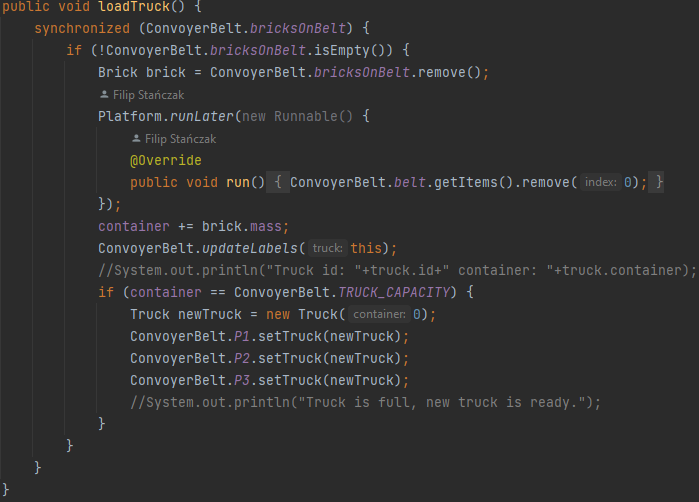


**Wykaz wyróżnionych punktów synchronizacji:**

Położenie cegły na taśmę produkcyjną:



Zdjęcie cegły z taśmy i załadowanie do ciężarówki:



**Wykaz obiektów synchronizacji:**

Semafor pozwalający na skorzystanie z taśmy produkcyjnej:

****

**Wykaz procesów sekwencyjnych:**

**-**Wątek główny sterujący całą aplikacją, powołujący do życia pracowników oraz wyświetlający symulację

-Wątki pracowników

**Listing programu:**

**Truck:**

package com.example.projektpw;  
  
import javafx.application.Platform;  
import javafx.scene.paint.Color;  
import javafx.scene.shape.Rectangle;  
  
public class Truck {  
 public int container;  
 public int id = 1;  
 public static int *nextId* = 1;  
 public Truck(int container) {  
 this.container = container;  
 this.id = *nextId*;  
 *nextId*++;  
 }  
 public void loadTruck() {  
 synchronized (Main.*bricksOnBelt*) {  
 if (!Main.*bricksOnBelt*.isEmpty()) {  
 Brick brick = Main.*bricksOnBelt*.remove();  
 Platform.*runLater*(new Runnable() {  
 @Override  
 public void run() {  
 Main.*belt*.getItems().remove(0);  
 }  
 });  
 container += brick.mass;  
 Main.*updateLabels*(this);  
 //System.out.println("Truck id: "+truck.id+" container: "+truck.container);  
 if (container == Main.*TRUCK\_CAPACITY*) {  
 Truck newTruck = new Truck(0);  
 Main.*P1*.setTruck(newTruck);  
 Main.*P2*.setTruck(newTruck);  
 Main.*P3*.setTruck(newTruck);  
 //System.out.println("Truck is full, new truck is ready.");  
 }  
 }  
 }  
 }  
}

**Convoyer Belt:**

package com.example.projektpw;  
  
import java.util.LinkedList;  
  
public class ConvoyerBelt {  
 public LinkedList<Brick> bricksList = new LinkedList<Brick>();  
 public synchronized void add(Brick brick){  
 bricksList.add(brick);  
 }  
 public synchronized Brick remove(){  
 return bricksList.removeFirst();  
 }  
 public Boolean isEmpty(){  
 return bricksList.isEmpty();  
 }  
 public int getSize(){  
 return bricksList.size();  
 }  
}

**Worker:**

package com.example.projektpw;  
  
import javafx.application.Platform;  
  
import static com.example.projektpw.Main.*bricksOnBelt*;  
  
public class Worker implements Runnable{  
 public String name;  
 public int brickType;  
 public int BELT\_CAPACITY;  
 public int BELT\_WEIGHT\_LIMIT;  
 public int TRUCK\_CAPACITY;  
 private int takeBrickTime;  
 public void setTruck(Truck truck) {  
 this.truck = truck;  
 }  
 private Truck truck;  
 public Worker(int brickType, Truck truck, int BELT\_CAPACITY, int BELT\_WEIGHT\_LIMIT, int TRUCK\_CAPACITY) {  
 this.brickType = brickType;  
 this.truck = truck;  
 this.BELT\_CAPACITY = BELT\_CAPACITY;  
 this.BELT\_WEIGHT\_LIMIT = BELT\_WEIGHT\_LIMIT;  
 this.TRUCK\_CAPACITY = TRUCK\_CAPACITY;  
 this.takeBrickTime = brickType\*100;  
 }  
 @Override  
 public void run() {  
 while (true) {  
 //System.out.println("-------"+HelloApplication.bricksOnBelt.size());  
 try {  
 Thread.*sleep*(takeBrickTime);  
 Main.*lock*.acquire();  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 layBrick(brickType);  
 Main.*lock*.release();  
 }  
 }  
 private void layBrick(int mass) {  
 synchronized (*bricksOnBelt*) {  
 int weight = 0;  
 for(Brick brick : *bricksOnBelt*.bricksList) {  
 weight += brick.mass;  
 }  
 //System.out.println("Belt weight: "+weight+" Worker P"+mass+" working..");  
 if (*bricksOnBelt*.getSize() < BELT\_CAPACITY &&  
 weight + mass <= BELT\_WEIGHT\_LIMIT &&  
 truck.container + weight + mass <= TRUCK\_CAPACITY) {  
 Brick brick = new Brick(brickType);  
 *bricksOnBelt*.add(brick);  
 Platform.*runLater*(new Runnable() {  
 @Override  
 public void run() {  
 Main.*belt*.getItems().add(String.*valueOf*(mass));  
 }  
 });  
 Main.*updateLabels*(truck);  
 //System.out.println("brick layed by P"+ brickType+", belt weight: "+(weight+mass)+"");  
 }else {  
 truck.loadTruck();  
 }  
 try{  
 Thread.*sleep*(200);  
 }  
 catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 }  
 }  
}

**Main:**

package com.example.projektpw;  
  
import javafx.application.Application;  
import javafx.application.Platform;  
import javafx.geometry.Insets;  
import javafx.geometry.Orientation;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.Label;  
import javafx.scene.control.ListView;  
import javafx.scene.layout.\*;  
import javafx.stage.Stage;  
  
import java.io.\*;  
import java.util.Properties;  
import java.util.concurrent.Semaphore;  
  
public class Main extends Application {  
 public static Semaphore *lock* = new Semaphore(1);  
 public Truck truck;  
 public static Worker *P1*;  
 public static Worker *P2*;  
 public static Worker *P3*;  
 public static int *BELT\_CAPACITY*;  
 public static int *BELT\_WEIGHT\_LIMIT*;  
 public static int *TRUCK\_CAPACITY*;  
 public static Label *truckLoadLabel* = new Label("Truck load: 0");  
 public static Label *truckCountLabel* = new Label("Truck count: 0");  
 public static Label *bricksOnBeltLabel* = new Label("Bricks on belt: 0");  
 private Label P1label = new Label("P1");  
 private Label P2label = new Label("P2");  
 private Label P3label = new Label("P3");  
 public static VBox *road* = new VBox();  
 public static ListView<String> *belt* = new ListView<>();  
 public static final ConvoyerBelt *bricksOnBelt* = new ConvoyerBelt();  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
  
 public void start(Stage primaryStage) {  
 readConfig();  
 primaryStage.setTitle("Symulator taśmy transportowej");  
  
 HBox root = new HBox();  
 root.setPadding(new Insets(10));  
 root.setSpacing(50);  
 root.setAlignment(Pos.*CENTER*);  
  
 HBox truckBoxMain = createTruckBoxMain();  
 VBox bricksOnBeltBoxMain = createBricksOnBeltBoxMain();  
 VBox workerBox = createWorkerBox();  
  
 root.getChildren().addAll(truckBoxMain, bricksOnBeltBoxMain, workerBox);  
  
 Scene scene = new Scene(root, 800, 200);  
 primaryStage.setScene(scene);  
 primaryStage.show();  
  
 Truck truck = new Truck(0);  
 *P1* = new Worker(1, truck, *BELT\_CAPACITY*, *BELT\_WEIGHT\_LIMIT*, *TRUCK\_CAPACITY*);  
 *P2* = new Worker(2, truck, *BELT\_CAPACITY*, *BELT\_WEIGHT\_LIMIT*, *TRUCK\_CAPACITY*);  
 *P3* = new Worker(3, truck, *BELT\_CAPACITY*, *BELT\_WEIGHT\_LIMIT*, *TRUCK\_CAPACITY*);  
 Thread p1 = new Thread(*P1*);  
 Thread p2 = new Thread(*P2*);  
 Thread p3 = new Thread(*P3*);  
  
 p1.start();  
 p2.start();  
 p3.start();  
 }  
  
 private HBox createTruckBoxMain(){  
 HBox box = new HBox();  
 *road*.setStyle("-fx-border-color: black; " +  
 "-fx-border-style: solid; " +  
 "-fx-border-width: 0 2px 0 2px;");  
 *road*.setMinSize(100, 100);  
 *road*.setAlignment(Pos.*CENTER\_RIGHT*);  
 VBox newBox = new VBox();  
 HBox truckCountBox = new HBox();  
 truckCountBox.setAlignment(Pos.*TOP\_LEFT*);  
 truckCountBox.getChildren().addAll(*truckCountLabel*);  
 HBox truckLoadBox = new HBox();  
 truckLoadBox.setAlignment(Pos.*BOTTOM\_LEFT*);  
 truckLoadBox.getChildren().addAll(*truckLoadLabel*);  
 newBox.getChildren().addAll(truckCountBox,truckLoadBox);  
 newBox.setSpacing(10);  
 newBox.setAlignment(Pos.*CENTER\_LEFT*);  
 box.setSpacing(20);  
 box.getChildren().addAll(newBox, *road*);  
 return box;  
 }  
  
 private VBox createBricksOnBeltBoxMain(){  
 VBox newBox = new VBox();  
 HBox beltBox = new HBox();  
 beltBox.setStyle("-fx-border-color: brown; " +  
 "-fx-border-style: solid; " +  
 "-fx-border-width: 2px; " +  
 "-fx-border-radius: 5px;");  
 beltBox.setAlignment(Pos.*BASELINE\_CENTER*);  
 beltBox.setMinSize(400, 100);  
 *belt*.setOrientation(Orientation.*HORIZONTAL*);  
 beltBox.getChildren().add(*belt*);  
 VBox beltLabel = new VBox();  
 beltLabel.setSpacing(50);  
 beltLabel.setAlignment(Pos.*BOTTOM\_CENTER*);  
 beltLabel.setPadding(new Insets(30));  
 beltLabel.getChildren().addAll(*bricksOnBeltLabel*);  
 newBox.getChildren().addAll(beltBox, beltLabel);  
 return newBox;  
 }  
  
 private VBox createWorkerBox(){  
 VBox newBox = new VBox();  
 newBox.setSpacing(50);  
 newBox.setAlignment(Pos.*CENTER\_RIGHT*);  
 newBox.getChildren().addAll(P1label, P2label, P3label);  
 return newBox;  
 }  
  
 private void readConfig(){  
 try (InputStream input = new FileInputStream("src/main/resources/config.properties")) {  
 Properties prop = new Properties();  
 prop.load(input);  
 *BELT\_CAPACITY* = Integer.*parseInt*(prop.getProperty("BELT\_CAPACITY"));  
 *BELT\_WEIGHT\_LIMIT* = Integer.*parseInt*(prop.getProperty("BELT\_WEIGHT\_LIMIT"));  
 *TRUCK\_CAPACITY* = Integer.*parseInt*(prop.getProperty("TRUCK\_CAPACITY"));  
  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
 public static void updateLabels(Truck truck) {  
 Platform.*runLater*(new Runnable() {  
 @Override  
 public void run() {  
 Main.*truckLoadLabel*.setText("Truck load: " + truck.container);  
 Main.*bricksOnBeltLabel*.setText("Bricks on belt: " + *bricksOnBelt*.getSize());  
 Main.*truckCountLabel*.setText("Truck count: " + (truck.id));  
 }  
 });  
 }  
}