
	<p style="text-align: center;">Politechnika Krakowska im. Tadeusza Kościuszki Wydział Fizyki, Matematyki i Informatyki</p>	
<p style="text-align: center;">Temat projektu: Zagadnienie szeregowania zadań cyklicznych. Implementacja algorytmu EDF (Earliest Deadline First)</p>		
<p style="text-align: center;">Miłosz Szlachetka Numer albumu: 114014</p>		

Spis treści

Wstęp	2
2.Projekt aplikacji	3
3.Technologie	5
4.Interfejs	6
5.Fragmenty kodu	11
6.Wnioski	13
7.Bibliografia	13

1. Wstęp

Celem niniejszej pracy jest projekt oraz implementacja algorytmu szeregowania terminowego EDF (Earliest Deadline First). Algorytm ten do szeregowania zadań wykorzystuje planowanie dynamiczne, gdzie plan przydziału zasobów dla określonego zadania jest ustalany w trakcie pracy systemu (na bieżąco). Algorytm EDF ustala kolejność wykonywania zadań na podstawie deadlineu. Do swojego działania wykorzystuje kolejkę priorytetową. Największy priorytet w kolejce otrzymuje zadanie, którego termin zakończenia (deadline) wypada najwcześniej. Następnie z kolejki tej będzie pobrane zadanie o najwyższym priorytecie i zostaną mu przydzielone zasoby niezbędne do jego wykonania. W algorytmie EDF stosowana jest strategia z wywłaszczaniem zadań. Oznacza to, że w momencie gdy w systemie pojawi się nowe zadanie, którego termin zakończenia wypada wcześniej niż wszystkich zadań znajdujących się aktualnie w systemie, system przerwie wykonywanie aktualnego zadania (wywłaszczy je) i zajmie się wykonywaniem nowoprzybyłego zadania. Z powodu stosowania wywłaszczania oraz tego, że algorytm przydziela zasoby dynamicznie, w czasie pracy systemu zmianom ulegają także priorytety zadań. Algorytm EDF wykorzystywany jest do szeregowania zadań cyklicznych, czyli takich, które mają się wykonywać co określoną ilość czasu. Wykazano, że algorytm ten jest optymalny, gdy używany jest pojedynczy procesor oraz stosowane jest wywłaszczanie, co oznacza, że jeśli istnieje poprawny harmonogram to EDF będzie działać poprawnie. Ponadto zestaw n zadań okresowych może być poprawnie planowany algorytmem EDF jeśli współczynnik wykorzystania procesora jest mniejszy lub równy 1 ($U \leq 1$). Współczynnik wykorzystania procesora wyraża się wzorem :

$$U = \sum_{i=1}^n \frac{C_i}{T_i} \leq 1,$$

C_i - czas wykonania i -tego zadania

T_i - okres wykonania i -tego zadania

Wadą algorytmu EDF jest niewątpliwie to, że nie jest on najprostszy w implementacji. Dodatkowo obciąża system bardziej niż algorytmy o szeregowaniu ze stałymi priorytetami. W wypadku gdy współczynnik wykorzystania procesora (U) jest większy niż 1 algorytm jest mało przewidywalny, a zadania nim szeregowane mogą ulegać coraz większym opóźnieniom (opóźnienie kumuluje się), co prowadzi do niewykonania ich w żądanym terminie.

2. Projekt aplikacji

Wymagania funkcjonalne

ID	Nazwa	Opis
WF-01	Dodawanie zadania	Dodawanie nowego zadania do uszeregowania. Nowo dodane zadanie pojawia się w tabeli wszystkich zadań.
WF-02	Generowanie zadań losowo	Generowanie określonej ilości zadań (wprowadzonej przez użytkownika) z losowo wybranym okresem oraz czasem wykonania zadania.
WF-03	Edycja zadania	Możliwość zmiany okresu lub czasu wykonania zadania.
WF-04	Usuwanie zadania	Usunięcie zadania z listy wszystkich zadań.
WF-05	Szeregowanie zadań	Rysowanie wykresu Gantta prezentującego harmonogram uszeregowanych zadań, które zostały uprzednio wprowadzone do aplikacji. Dodatkowo obliczane i wyświetlane są: współczynnik wykorzystania procesora (U) oraz NWW (najmniejsza wspólna wielokrotność) po to by użytkownik wiedział ile wynosi okres dla całego harmonogramu.
WF-06	Odczyt z pliku	Wprowadzanie zadań do uszeregowania z pliku tekstowego (txt)

Wymaganie нефункционалне

ID	Nazwa	Opis
WNF-01	Rodzaj aplikacji	Aplikacja jest aplikacją desktopową, którą uruchamia się przy użyciu pliku .jar. Aby aplikacja działała poprawnie, na komputerze musi być zainstalowana Java.
WNF-02	Interfejs	Aplikacja posiada graficzny interfejs użytkownika z moduleм rysującym wykres oraz pole, które wyświetla współczynnik wykorzystania procesora (U).
WNF-03	Limit zadań	Maksymalna liczba zadań jakie może uszeregować użytkownik wynosi 10.
WNF-04	Zakres wartości na wykresie	Wykres Gantta zawiera dwie osie: pionową i poziomą. Oś pionowa zawiera ID zadań. Oś pozioma jest osią czasu. Jej zakres wynosi od 0 do maksymalnie 399.
WNF-05	Wprowadzanie wartości	Dane wprowadzane przez użytkownika muszą być wartościami całkowitymi, dodatnimi. W przeciwnym wypadku pojawia się monit ze stosownym komunikatem.

3.Technologie

Aplikacja jest aplikacją desktopową, którą uruchamia się za pomocą pliku wykonywalnego EDF.jar . Do implementacji aplikacji wykorzystano następujące technologie:

Java - obiektowy język programowania wysokiego poziomu. Oferuje wiele zaawansowanych konstrukcji programistycznych takich jak np: typy generyczne, obsługa wyjątków. Język java zwalnia programistę z odpowiedzialności zwalniania zasobów, dzięki mechanizmowi garbage collector. Programy pisane w języku java posiadają własność przenośności, co oznacza, że raz napisany program będzie funkcjonował poprawnie na wielu środowiskach uruchomieniowych. Dzieje się tak dlatego, że kod javy jest kompilowany do kodu pośredniego (bytecode), który może być wykonywany na dowolnym systemie, który dostarcza wirtualnej maszyny Java.

Technologia java pozwala tworzyć wiele rodzajów aplikacji, takich jak aplikacje desktopowe czy też webowe oraz mobilne.

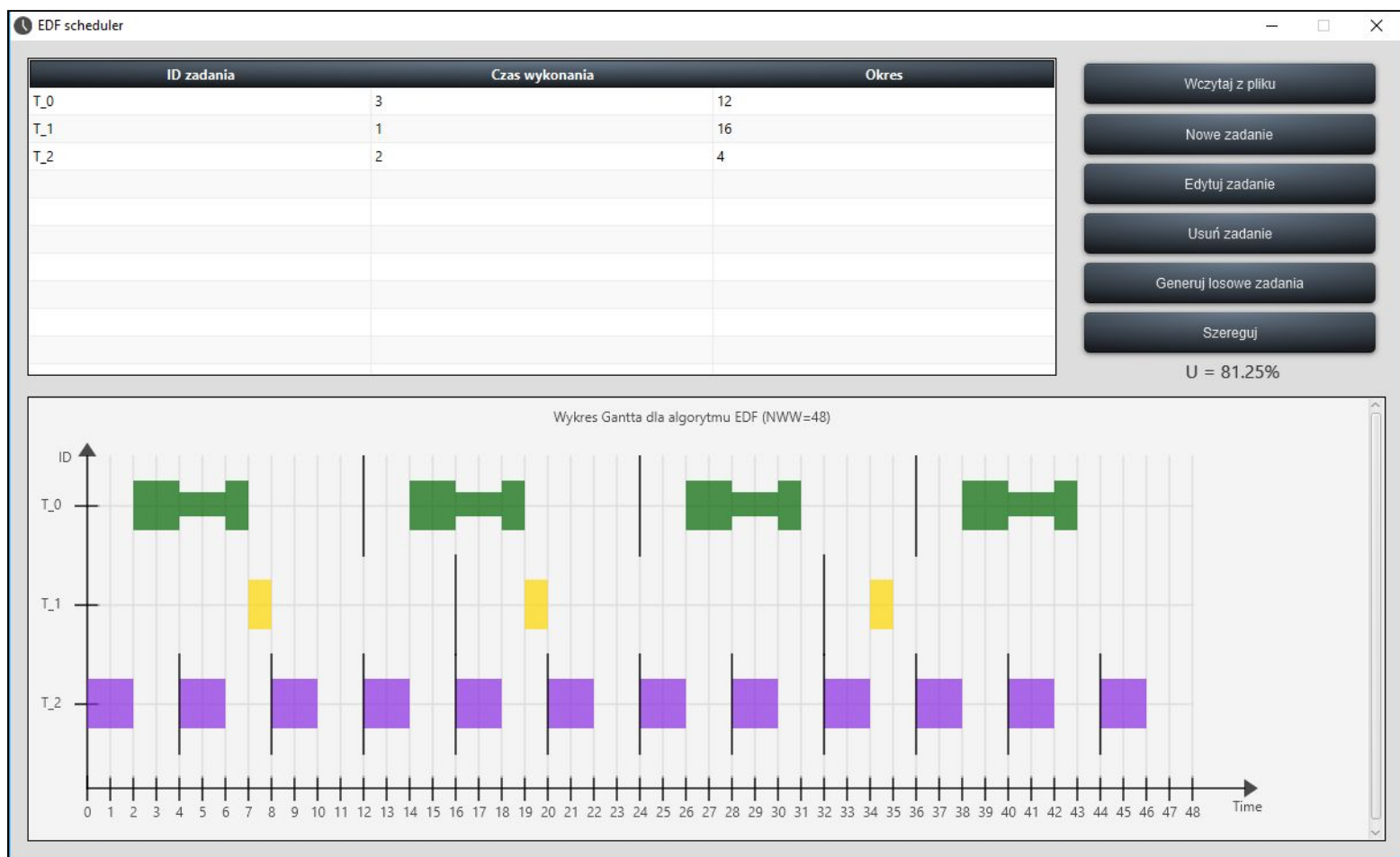
JavaFX - biblioteka javy umożliwiająca tworzenie rozbudowanych interfejsów graficznych zarówno dla aplikacji desktopowych jak i internetowych. JavaFX jest obecnie standardem tworzenia GUI oraz następcą biblioteki Swing. Pozwala tworzyć bardziej rozbudowany i bardziej estetyczny interfejs niż oferowała starsza technologia Swing. Dzięki zastosowaniu technologii JavaFX kod widoku (interfejs) aplikacji jest definiowany w języku XML, co pozwala odseparować go od kodu Javy, co z kolei jednoznacznie odseparowuje logikę od warstwy prezentacji.

CSS - czyli kaskadowe arkusze stylów. Język służący do definiowania wyglądu aplikacji. Opisuje sposób wyświetlania elementów graficznego interfejsu użytkownika. Dostarcza odpowiednich dyrektyw, które definiują stylistykę elementu interfejsu użytkownika. CSS daje możliwość odseparowania tego co ma być zawarte w interfejsie użytkownika od tego w jaki sposób mają zostać wyświetlone jego poszczególne elementy. Dzięki tej technologii mamy możliwość ustawiania np. położenia, rozmiaru, koloru, cieni, ramek itp. poszczególnych składowych interfejsu graficznego.

4. Aplikacja

Aplikacja posiada graficzny interfejs użytkownika, który został zaimplementowany przy użyciu technologii JavaFX oraz CSS. Uruchomienie aplikacji następuje po dwukrotnym kliknięciu ikony o nazwie “EDF_Projekt”. Do poprawnego działania programu wymagane jest posiadanie oprogramowania Java.

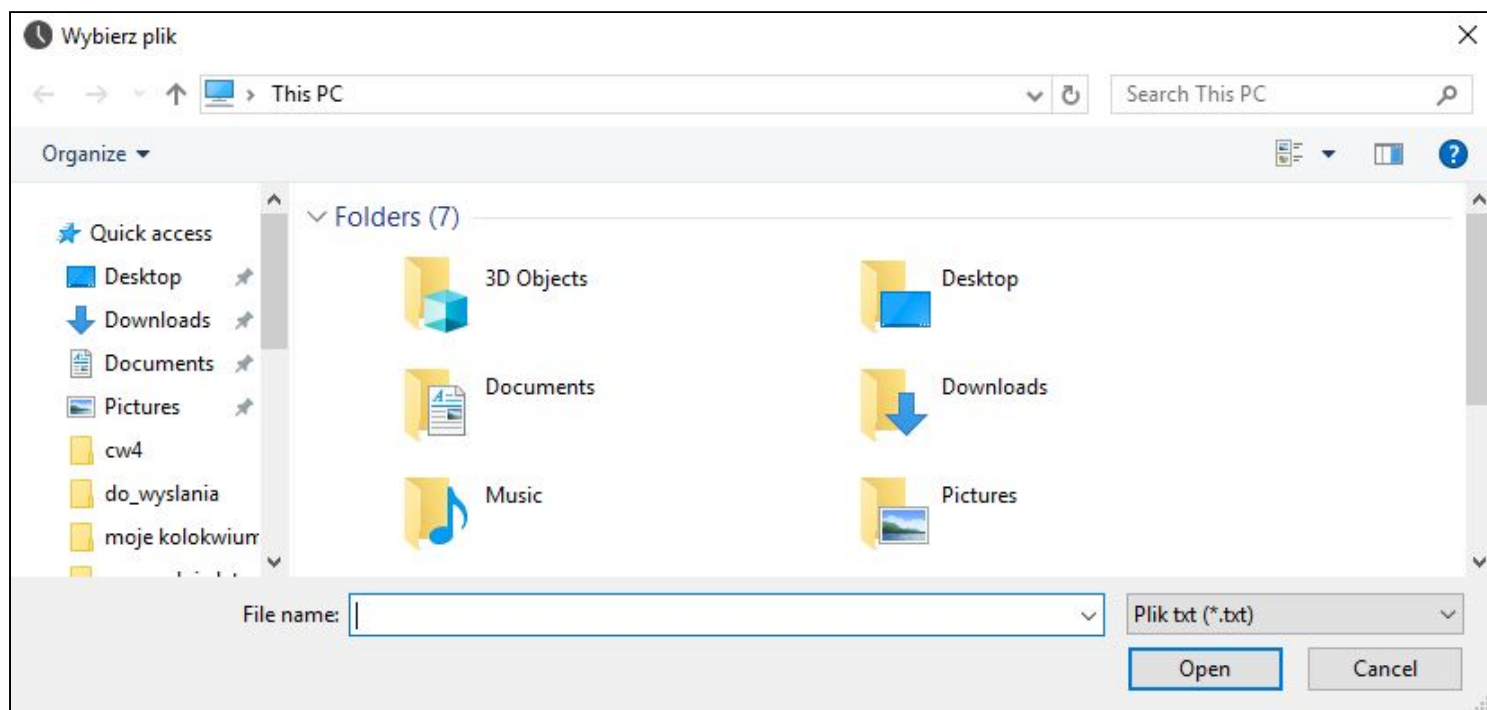
Poniżej znajduje się zrzut ekranu, prezentujący ekran główny aplikacji, który jest widoczny jako pierwszy po uruchomieniu aplikacji. Składa się on z 3 części. Po lewej stronie znajduje się tabela zawierająca zestaw zadań, które mają zostać uszeregowane. Każde zadanie w tej tabeli może zostać podświetlone dla lepszej widoczności. Podświetlenie następuje po kliknięciu na zadanie lewym klawiszem myszy. Dodatkowo tabela ta ma możliwość zmiany układu kolumn. Klikając na nazwę kolumny i przytrzymując klawisz, można przeciągać kolumnę w odpowiednie miejsce. Prawa część interfejsu zawiera menu, na które składa się pięć przycisków. Pod przyciskiem “Szereguj” wyświetlany jest procentowy wskaźnik wykorzystania procesora (U). Jest on widoczny po użyciu przycisku “Szereguj”. U dołu ekranu widoczny jest wykres Gantta, który przedstawia harmonogram (uszeregowanie) zadań. Jeśli wykres Gantta będzie większy niż szerokość okna w którym się wyświetla, to pojawi się scrollbar, który umożliwi przesuwanie wykresu w prawo lub lewo oraz w górę i dół.



[Rys. 1] Główny interfejs użytkownika

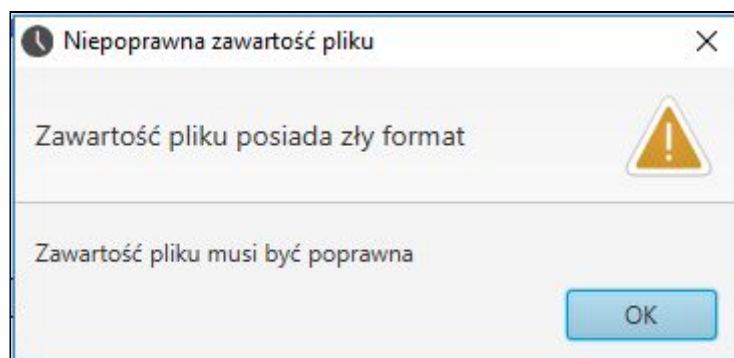
Przyciski znajdujące się w menu widocznym po prawej stronie ekranu służą do wykonywania funkcji, jakie oferuje program. Przycisk "Wczytaj z pliku" służy do wprowadzania z pliku tekstowego (*.txt) do programu nowych zadań do uszeregowania. Po jego kliknięciu wyświetlane jest okno wyboru pliku, które zostało zaprezentowane na Rys.2. Aby plik zawsze miał format tekstowy została zablokowana możliwość wybrania innego pliku niż tekstowy (txt). Aby poprawnie przetworzyć jego zawartość należy przechowywać w nim dane w następującej postaci: Każde zadanie definiowane jest w osobnej linii. Pierwsza liczba w linii to czas wykonania zadania. Następnie musi wystąpić jedna lub więcej spacji. Druga liczba to czas okresu wykonania zadania. Przykład zdefiniowania trzech zadań:

```
1  12
3  7
7    45
```



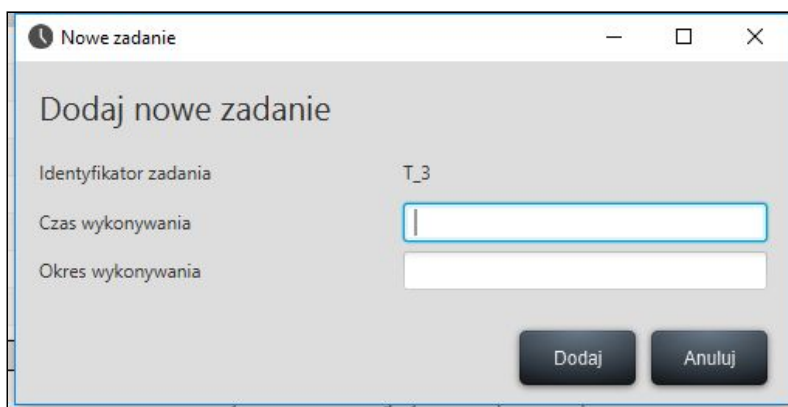
[Rys.2] Okno wyboru pliku

W przypadku gdy wskazany plik posiada niepoprawną zawartość (niezgodną z wyżej opisanymi wymaganiami) pojawia się komunikat o błędzie (Rys.3), a zadania nie są wczytywane.



[Rys.3] Kominikat o niepoprawnej zawartości pliku

Przycisk “Nowe zadanie”, służy do dodawania nowego zadania do tabeli przechowującej wszystkie zadania. Jego kliknięcie powoduje wyświetlenie się poniższego okna (Rys.2). Aby dodać nowe zadanie wystarczy wpisać jego czas oraz okres wykonania i zatwierdzić operację przyciskiem Dodaj. W przypadku chęci zrezygnowania z dodawania zadania należy użyć przycisku Anuluj.



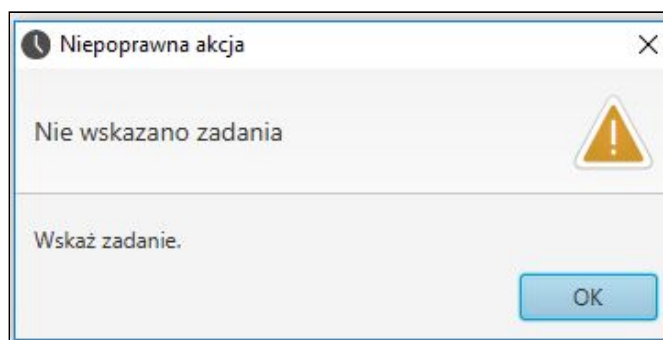
[Rys. 4] Dodawanie zadania.

Każde zadanie w tabeli może zostać edytowane. Edycji podlega czas oraz okres wykonania zadania. Identyfikator jest niezmienny. W celu edycji zadania należy wskazać je w tabeli oraz użyć przycisku “Edytuj zadanie”. Powoduje to otwarcie nowego okna, które prezentuje się następująco.



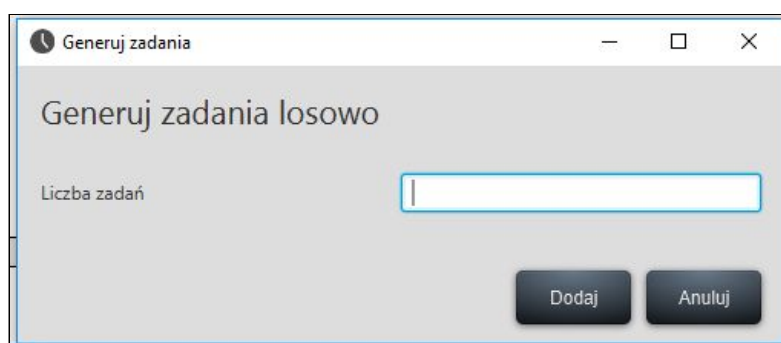
[Rys. 5] Edycja zadania

Usuwanie zadania następuje gdy użytkownik, podobnie jak przy edycji zadania, wskaże zadanie na liście oraz użyje przycisku “Usuń zadanie”. W przypadku niewskazania zadania, podczas edycji lub usuwania, pojawia się poniższy komunikat.



[Rys. 6] Komunikat błędu - nie wskazano zadania

Generacja losowych zadań jest możliwa po użyciu przycisku “Generuj losowe zadania”. Po jego użyciu pojawia się okno, w którym należy podać ilość zadań do generacji. Zadania te mają tę cechę, że czas wykonania każdego z zadań będzie zawsze mniejszy niż okres. Poniżej znajduje się omawiane okno generacji losowych zadań.



[Rys. 7] Losowa generacja zadań

Ostatnim przyciskiem z menu jest przycisk “Szereguj”. Po kliknięciu w niego następuje uszeregowanie zadań znajdujących się w tabeli, narysowanie wykresu Gantta prezentującego harmonogram uszeregowania oraz wyliczenie i wyświetlenie współczynnika wykorzystania procesora.

5. Fragmenty kodu

Struktura danych (klasa) reprezentująca zadanie w systemie. Zadanie posiada czas wykonania, okres oraz identyfikator, a także zmienną “counter”, która służy do generowania unikalnych identyfikatorów zadań.

```
1 public class Task {  
2  
3     private static int counter = 0;  
4  
5     private SimpleIntegerProperty executionTime;  
6     private SimpleIntegerProperty period;  
7     private SimpleStringProperty id;  
8  
9     public Task(int executionTime, int period) {  
10         this();  
11         this.executionTime = new SimpleIntegerProperty(executionTime);  
12         this.period = new SimpleIntegerProperty(period);  
13     }  
14  
15     public Task(){  
16         this.id = new SimpleStringProperty("T_"+counter++);  
17     }  
18  
19     public int getExecutionTime() {  
20         return executionTime.get();  
21     }  
22  
23     public void setExecutionTime(int executionTime) {  
24         this.executionTime = new SimpleIntegerProperty(executionTime);  
25     }  
26  
27     public int getPeriod() {  
28         return this.period.get();  
29     }  
30  
31     public void setPeriod(int period) {  
32         this.period=new SimpleIntegerProperty(period);  
33     }  
34  
35     public String getId() {  
36         return id.get();  
37     }  
38 }
```

[Rys. 7] Struktura danych (klasa) reprezentująca zadanie w systemie

Poniżej znajduje się fragment kodu programu w języku Java realizujący szeregowanie zadań zgodnie z algorytmem EDF.

```
1 static List<Task> schedule(final List<Task> taskList, int LCM) {
2
3     List<Task> orderedTasks = new LinkedList<>();
4     Map<Integer, List<Task>> waitingMap = new HashMap<>();
5
6     for (int timeUnit = 0; timeUnit < LCM; timeUnit++) {
7
8         for (Task t : taskList) {
9             if (timeUnit % t.getPeriod() == 0) {
10
11                 if (!waitingMap.containsKey(timeUnit + t.getPeriod())) {
12                     waitingMap.put(timeUnit + t.getPeriod(), new LinkedList<>());
13                 }
14
15                 for (int i = 0; i < t.getExecutionTime(); i++) {
16                     waitingMap.get(timeUnit + t.getPeriod()).add(t);
17                 }
18             }
19         }
20
21         if (!waitingMap.isEmpty()) {
22             int minKey = Collections.min(waitingMap.keySet());
23             orderedTasks.add(waitingMap.get(minKey).remove(0));
24             if (waitingMap.get(minKey).isEmpty()) {
25                 waitingMap.remove(minKey);
26             }
27         } else {
28             orderedTasks.add(null);
29         }
30     }
31
32     return orderedTasks;
33 }
```

[Rys. 8] Kod realizujący szeregowanie zadań według algorytmu EDF.

Fragment programu wyznaczający współczynnik wykorzystania procesora dla listy zadań.

```
1 static BigDecimal calculateUtilizationPercentage(final List<Task> taskList) {
2
3     int LCM = getLeastCommonMultiple(taskList);
4     int sum = calculateSumOfTime(taskList, LCM);
5
6     BigDecimal U = new BigDecimal(sum * 100).divide(new BigDecimal(LCM), 2, RoundingMode.HALF_UP);
7     return U;
8 }
```

[Rys. 9] Kod realizujący wyznaczanie współczynnika wykorzystania procesora

6.Wnioski

Zaimplementowana w ramach projektu aplikacja realizuje szeregowanie zadań cyklicznych zgodnie z algorytmem EDF. Do przedstawienia harmonogramu uszeregowania wykorzystano technologię JavaFX oraz komponent Canvas. Wykres jest rysowany bez użycia żadnych gotowych, dodatkowych bibliotek do generacji wykresu Gantta. Wybór technologii JavaFX oraz CSS pozwolił na wykonanie aplikacji, która posiada przejrzysty i wygodny w użytkowaniu interfejs. Użycie języka Java dało możliwość skutecznej implementacji algorytmu zgodnie z paradygmatem programowania obiektowego. Ważnym aspektem jest również to, że aplikacja jest przenośna (może być uruchamiana na dowolnym systemie operacyjnym).

7.Bibliografia

- [1]<https://docs.oracle.com/javase/8/javase-clienttechnologies.htm>
- [2]<https://docs.oracle.com/javase/7/docs/api/>
- [3]<https://docs.oracle.com/javase/8/docs/>
- [4]<http://retis.sssup.it/~lipari/courses/rtos/lucidi/edf.pdf>
- [5]<http://retis.sssup.it/~lipari/courses/stro6/10.edf.pdf>
- [6]<https://docs.oracle.com/javafx/2/canvas/jfxpub-canvas.htm>
- [7]<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/canvas/Canvas.html>
- [8]http://sequoia.ict.pwr.wroc.pl/~witold/sicr/sicr_sched_s.pdf
- [9]<http://jedrzej.ulasiewicz.staff.iiar.pwr.wroc.pl/KomputeroweSystSter/wyklad/SzeregowanieRTS-8.pdf>