# OOR Ćwiczenie 4 – Programowanie asynchroniczne

Autor: Łukasz Pawłowski

Grupa: 125NCI\_B

Link do projektu: <https://github.com/mojzesz/oor-lab-4>

Język programowania: C#

**Zadania do zrealizowania:**

**OOR ćw. 4**

1. Za pomocą wybranego przez siebie języka programowania zademonstruj w praktyce ideę programowania asynchronicznego
2. Bądź gotowy do prezentacji możliwości i modyfikacji napisanego programu.

**Przykładowe pytania OOR ćw. 4**

* Wyjaśnij ideę programowania asynchronicznego.

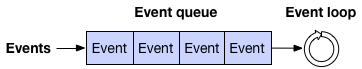
Programowanie asynchroniczne polega na utworzeniu kilku wątków które zajmują się obliczeniami, oczekiwaniem na zasoby lub innymi zadaniami równocześnie, podczas gdy przez główny wątek cały czas jest możliwość dostępu do responsywnego UI. Wątki są niezależne od siebie ani od wyników swojej pracy (ewentualnie do momentu synchronizacji wyników).

* Wyjaśnij różnicę między komunikacją synchroniczną a asynchroniczną.

W komunikacji synchronicznej czekamy na odpowiedź wątku – czekamy, aż się wykona, blokując inne wątki. Interface jest wtedy blokowany. W komunikacji asynchronicznej nie potrzeba czekać na wykonanie wątku.

* Wyjaśnij pojęcie 'event loop'. Podaj przykład.

Event loop jest odpowiedzialny za pobieranie kolejnych elementów z event queue oraz ich wywoływanie. W kolejne mogą się znajdować różne wydarzenia: click, upload, open itd. Generalnie jest to twór, który czeka na przyjmowanie i wywoływanie kolejnych wydarzeń. W programie może istnieć wiele takich pętli z których najważniejsza i o największym priorytecie jest pętla *main*.



Przykładowo proces okna głównego aplikacji może czekać na wciśnięcie któregoś przycisku przez użytkownika, aby wywołać przypisaną do niego funkcję.

Procesy mogą być zakleszczone (deadlock) co oznacza, że procesy czekają jeden na drugiego na zwolnienie zasobów, ich stan się nie zmienia; livelock – co oznacza, że procesy czekają na kilka zasobów, zmieniają swój stan, zmieniają stan zajętych zasobów, ale nigdy nie mają wszystkich potrzebnych do wykonania się; zagłodzenie (starvation) – proces czeka na zasób, jednak jest on cały czas zajęty i proces nigdy nie uzyskuje dostępu do oczekiwanego zasobu.

# Realizacja zadań

Napisany program ma za zadanie załadować jednocześnie 5 stron internetowych, z których jedna ze stron posiada błędny adres. Mimo, że jedna z nich posiada błędny adres i jest wywoływana jako pierwsza, pozostałe strony załadują się bez problemu. Na dole aplikacji pokazane są dwa liczniki: jeden wskazuje już otwarte strony, drugi natomiast strony otwierane. Dodatkowo do każdej zakładki dopisywany jest tekst – czy strona w danym momencie jest otwierana czy już otwarta. Metoda getStringAsynch pozwala na asynchroniczne pobranie zawartości strony, po czym zwiększany jest licznik pobieranych stron, następnie wywoływany jest await który czeka na zakończenie pobrania i na końcu wywoływana jest funkcja dodająca kolejny numer otwartej strony.