Wątki w programowaniu to sekwencje instrukcji, które mogą być wykonywane równolegle w ramach jednego procesu. Wielowątkowość pozwala programom na równoległe wykonywanie wielu zadań, co przyspiesza czas odpowiedzi aplikacji oraz zwiększa wydajność systemu.

Wątki są często wykorzystywane w aplikacjach, które muszą wykonywać wiele operacji jednocześnie, takich jak przetwarzanie danych, komunikacja sieciowa, obsługa interfejsu użytkownika, itp. Wątki pozwalają na rozdzielenie złożonych zadań na mniejsze części i wykonanie ich równolegle, co przyspiesza cały proces.

Prosty przykład kodu w Javie uruchamiający dodatkowo 2 wątki.

public class ThreadExample **{**

public static void main**(**String**[]** args**)** **{**

// Tworzymy nowe wątki

Thread thread1 **=** **new** Thread**(new** Task**(**"Wątek 1"**));**

Thread thread2 **=** **new** Thread**(new** Task**(**"Wątek 2"**));**

// Uruchamiamy wątki

thread1**.**start**();**

thread2**.**start**();**

**}**

**}**

class Task **implements** Runnable **{**

private String name**;**

public Task**(**String name**)** **{**

**this.**name **=** name**;**

**}**

public void run**()** **{**

**for** **(**int i **=** 1**;** i **<=** 10**;** i**++)** **{**

System**.**out**.**println**(**name **+** ": " **+** i**);**

**try** **{**

// Oczekiwanie na krótki czas przed wykonaniem kolejnej iteracji pętli

Thread**.**sleep**(**1000**);**

**}** **catch** **(**InterruptedException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

**}**

Na wątki można też poczekać w wątku głównym:

public class ThreadExample2 **{**

public static void main**(**String**[]** args**)** **throws** InterruptedException **{**

// Tworzymy nowe wątki

Thread thread1 **=** **new** Thread**(new** Task**(**"Wątek 1"**));**

Thread thread2 **=** **new** Thread**(new** Task**(**"Wątek 2"**));**

// Uruchamiamy wątki

thread1**.**start**();**

thread2**.**start**();**

// Oczekiwanie na zakończenie pracy pozostałych wątków

thread1**.**join**();**

thread2**.**join**();**

// Wyświetlanie komunikatu po zakończeniu pracy wszystkich wątków

System**.**out**.**println**(**"Praca wszystkich wątków zakończona."**);**

JOptionPane**.**showMessageDialog**(null,** "Praca wszystkich wątków zakończona."**);**

**}**

**}**

class Task **implements** Runnable **{**

private String name**;**

public Task**(**String name**)** **{**

**this.**name **=** name**;**

**}**

public void run**()** **{**

**for** **(**int i **=** 1**;** i **<=** 10**;** i**++)** **{**

System**.**out**.**println**(**name **+** ": " **+** i**);**

**try** **{**

// Oczekiwanie na krótki czas przed wykonaniem kolejnej iteracji pętli

Thread**.**sleep**(**1000**);**

**}** **catch** **(**InterruptedException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

**}**

Wątki można też tworzyć w pętli:

public class ThreadExample3 **{**

public static void main**(**String**[]** args**)** **{**

**for** **(**int i **=** 1**;** i **<=** 3**;** i**++)** **{**

Thread t **=** **new** Thread**(new** WorkerThread**(**i**));**

t**.**start**();**

**}**

**for** **(**int i **=** 1**;** i **<=** 10**;** i**++)** **{**

System**.**out**.**println**(**"Wątek główny: " **+** i**);**

**}**

**}**

private static class WorkerThread **implements** Runnable **{**

private int threadNum**;**

public WorkerThread**(**int threadNum**)** **{**

**this.**threadNum **=** threadNum**;**

**}**

@Override

public void run**()** **{**

**for** **(**int i **=** 1**;** i **<=** 10**;** i**++)** **{**

System**.**out**.**println**(**"Wątek " **+** threadNum **+** ": " **+** i**\***i**);**

**}**

**}**

**}**

**}**

Poniżej prosty program w Javie, który tworzy dwa wątki, które wyświetlają liczby od 1 do 10.

public class ThreadExample1 **{**

public static void main**(**String**[]** args**)** **{**

// Tworzenie wątku

Thread t1 **=** **new** Thread**(new** Runnable**()** **{**

@Override

public void run**()** **{**

**for** **(**int i **=** 1**;** i **<=** 10**;** i**++)** **{**

System**.**out**.**println**(**"Wątek 1: " **+** i**);**

**}**

**}**

**});**

// Uruchamianie wątku

t1**.**start**();**

// Tworzenie drugiego wątku

Thread t2 **=** **new** Thread**(new** Runnable**()** **{**

@Override

public void run**()** **{**

**for** **(**int i **=** 1**;** i **<=** 10**;** i**++)** **{**

System**.**out**.**println**(**"Wątek 2: " **+** i**);**

**}**

**}**

**});**

// Uruchamianie drugiego wątku

t2**.**start**();**

**}**

**}**

1. Napisz program, który stworzy dwie duże tablice liczb i utwórz dwa wątki, które będą sumować elementy tablicy w sposób równoległy. Porównaj czas wykonania z użyciem jednego wątku i dwóch wątków. Następnie porównaj wynik dla kilku wątków – można użyć „Runtime.getRuntime().availableProcessors()” aby otrzymać liczbę rdzeni procesora.

Do realizacji zadania można użyć poniższego kodu:

**import** java**.**util**.**Random**;**

public class ParallelSum **{**

private static final int ARRAY\_SIZE **=** 100000000**;**

private static final int THREAD\_COUNT **=** 2**;**

private static int**[]** generateArray**(**int size**)** **{**

Random rand **=** **new** Random**();**

int**[]** arr **=** **new** int**[**size**];**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)** **{**

arr**[**i**]** **=** rand**.**nextInt**(**10**);**

**}**

**return** arr**;**

**}**

private static long sum**(**int**[]** arr**)** **{**

long result **=** 0**;**

**for** **(**int i **=** 0**;** i **<** arr**.**length**;** i**++)** **{**

result **+=** arr**[**i**];**

**}**

**return** result**;**

**}**

private static class SumThread **implements** Runnable **{**

private int**[]** arr**;**

private int startIndex**;**

private int endIndex**;**

private long result**;**

public SumThread**(**int**[]** arr**,** int startIndex**,** int endIndex**)** **{**

**this.**arr **=** arr**;**

**this.**startIndex **=** startIndex**;**

**this.**endIndex **=** endIndex**;**

**}**

public long getResult**()** **{**

**return** result**;**

**}**

@Override

public void run**()** **{**

**for** **(**int i **=** startIndex**;** i **<** endIndex**;** i**++)** **{**

result **+=** arr**[**i**];**

**}**

**}**

**}**

public static void main**(**String**[]** args**)** **throws** InterruptedException **{**

int**[]** arr1 **=** generateArray**(**ARRAY\_SIZE**);**

int**[]** arr2 **=** generateArray**(**ARRAY\_SIZE**);**

long startTime **=** System**.**currentTimeMillis**();**

long result1 **=** sum**(**arr1**);**

long elapsedTime1 **=** System**.**currentTimeMillis**()** **-** startTime**;**

System**.**out**.**println**(**"Elapsed time with one thread: " **+** elapsedTime1 **+** "ms"**);**

System**.**out**.**println**(**"Result with one thread: " **+** result1**);**

startTime **=** System**.**currentTimeMillis**();**

// ...

long elapsedTime2 **=** System**.**currentTimeMillis**()** **-** startTime**;**

System**.**out**.**println**(**"Elapsed time with " **+** THREAD\_COUNT **+** " threads: " **+** elapsedTime2 **+** "ms"**);**

System**.**out**.**println**(**"Result with " **+** THREAD\_COUNT **+** " threads: " **+** result2**);**

**}**

**}**

1. Przetestuj a następnie zmień poniższy program tak, aby pobierał kilka plików z różnych serwerów jednocześnie.

public class FileDownloader **{**

public static void downloadFile**(**String fileUrl**,** String saveFilePath**)** **throws** IOException **{**

URL url **=** **new** URL**(**fileUrl**);**

**try** **(**BufferedInputStream in **=** **new** BufferedInputStream**(**url**.**openStream**());**

FileOutputStream fileOutputStream **=** **new** FileOutputStream**(**saveFilePath**))** **{**

byte**[]** dataBuffer **=** **new** byte**[**1024**];**

int bytesRead**;**

**while** **((**bytesRead **=** in**.**read**(**dataBuffer**,** 0**,** 1024**))** **!=** **-**1**)** **{**

fileOutputStream**.**write**(**dataBuffer**,** 0**,** bytesRead**);**

**}**

**}**

**}**

public static void main**(**String**[]** args**)** **{**

String fileUrl **=** "https://scieżka-do-pliku"**;**

String saveFilePath **=** "sciezka na dysku.txt"**;**

**try** **{**

downloadFile**(**fileUrl**,** saveFilePath**);**

System**.**out**.**println**(**"Plik zapisany."**);**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

1. Napisz program, który będzie przyjmował ścieżki katalogów i dla każdego katalogu w oddzielnym wątku wyświetli listę plików wraz z katalogami oraz policzy zajmowane miejsce na dysku.