Klasa Thread

- Metody
  - xxPriority()
  - xxxName()
  - setDaemon
  - join() / run()
  - Interrupt() / isInterrupted()
  - setDaemon / getThreadGroup()
- Metody statyczne
  - CurrentThread()
  - Interrupted()
  - sleep() / yield()

Zadanie 1:

Napisać program, który pobiera jako pierwszy parametr nazwę pliku oraz jako następne parametry katalogi.

Dla każdego podanego katalogu tworzy osobny wątek, w którym przeszukuje zawartość katalogu w celu znalezienia podanego pliku.

Zadanie 2:

- Rozszerzona wersja zadania 1
  - Przeszukiwanie rekurencyjne
  - Wątki trzymane w generyku
  - Wyszukiwanie po wyrażeniu regularnym
  - Wątek nie szuka w katalogach przeszukanych przez inne wątki

Zadanie 3:

- Prosta klasa licznik
  - Operacja zwiększania wartości licznika
  - Współdzielona pomiędzy wątki
  - Synchronizacja

„Niezmienne” obiekty

- Brak setter'ów
- Wszystkie pola final i private
- Nie pozwól, aby klasy dziedziczące po niezmiennej klasie nadpisywały metody. Sposób pierwszy – prostszy: zadeklarować klasę jako final. Sposób drugi: zadeklarować konstruktor jako prywatny oraz stworzyć fabrykę.
- Jeśli pola klasy są referencjami do „zmiennych” obiektów, nie pozwól tym obiektom zmienić swój stan:
  - Brak metod mogących zmieniać „zmienne” obiekty.
  - Nie współdziel referencji do „zmiennych” obiektów. Nigdy nie przechowuj referencji do zewnętrznych obiektów pobieranych przez konstruktor, jeśli to konieczne twórz kopie obiektów oraz przechowuj do nich referencje.

Synchronizowane typy generyczne

- **BlockingQueue** - FIFO, blokuje wątek kiedy dodajemy do pełnej kolejki albo czytamy z pustej.
- **ConcurrentMap** – interface dziedziczący po java.util.Map, zapewnia użyteczne operacje atomowe, usuwanie albo zamiana par klucz-wartość jeśli klucz jest dostępny, albo dodawanie pary klucz-wartość jesli klucza nie ma. Jedna z implementacji tego interface'u jest **ConcurrentHashMap** – współbieżny odpowiednik **HashMap.**
- **ConcurrentNavigableMap** interface dziedziczący po **ConcurrentMap** który zapewnia aproksymacyjne trafienia. Jedna z implementacji tego interface'u jest **ConcurrentSkipListMap** – współbieżny odpowiednik **TreeMap**.

Interface'y Executor

- **Executor** Interface: zapewnia pojedynczą metodę **Execute** która przyjmuje obiekt **Runnable** oraz w zależności od implementacji albo uruchamia nowy wątek, albo przesyła zadanie do istniejącego wątku, lub dodaje zadanie do kolejki zadań.
- **ExecutorService** Interface: zapewnia metodę **submit** która przyjmuje obiekt **Runnable** oraz **Callable**. **Submit** zwraca wartość, oraz możliwe jest przekazywanie dużych kolekcji zdań.
- **ScheduledExecutorService** Interface: dziedziczy po interface'sie **ExecutorService**, zapewnia jego metody z możliwości rozmieszczenia w czasie.

Zadanie 4:

- Modyfikacja zadania 2
  - Zamiana listy wątków na obiekt implementujący jeden z interfac'ów Executor
  - Porównanie czasu wyszukiwania

Zadanie 5:

- Modyfikacja zadania 3
  - Zmiana typów prostych na typy atomowe