Podstawy

1. Napisz program, w którym klasa `HelloWorld` implementuje wątek poprzez dziedziczenie z klasy `Thread`. Zadaniem wątku będzie wyświetlenie na ekranie napisu „Hello, world". Wątek główny programu po uruchomieniu wątku `HelloWorld` powinien wyświetlić na ekranie napis „Koniec".

   - Uruchom program kilka razy, czy kolejność napisów jest taka sama, czy zmienia się?
   - Zmodyfikuj program tak, aby utworzył 3 wątki `HelloWorld` – każdemu nadaj unikalny identyfikator w konstruktorze klasy, tak aby wątek wyświetlał napis „Hello, world [id]", gdzie [id] jest identyfikatorem wątku. Uruchom program kilka razy, czy napisy zawsze wyświetlają się w tej samej kolejności?
   - Dodaj klasę `HelloWorldRunnable`, której działanie jest analogiczne do `HelloWorld`, ale implementuje interfejs `Runnable` zamiast dziedziczenia z klasy `Thread`.
   - W wątku głównym dodaj oczekiwanie na zakończenie działania wątków roboczych za pomocą metody `Thread.join`.

2. Napisz program, w którym działają 2 wątki obliczeniowe korzystające ze wspólnego licznika implementowanego przez klasę `Licznik`. Wątki implementowane są przez klasę `Wykonawca`. Zadaniem wątków jest n-krotne (liczba n podana jest w konstruktorze) dodawanie do licznika wartości `delta` ustalonej w konstruktorze (1 dla wątku nr 1, -1 dla wątku nr 2).

   Wątek główny powinien poczekać na zakończenie działania wątków (metoda `join` z klasy `Thread`), a następnie wyświetlić na ekranie końcową wartość licznika.

   Klasa `Licznik` ma:

   - prywatne pole `int wartosc`, które przechowuje aktualną wartość licznika,
   - metodę `void modyfikuj(int n)`, która dodaje n do wartości licznika,
   - metodę `int dajWartosc()`, która zwaraca bieżącą wartość licznika.

   Następnie:

   a) Wykonać program z wartościami $n \in 10^3, 10^5, 10^7$, jakie są różnice w wynikach?

b) W pętli obliczeniowej w klasie `Wykonawca` dodać usypianie wątku na 0 ms. Jak to wpływa na zachowanie programu?

c) Pole `wartosc` w klasie `Licznik` zmienić na ulotne (`volatile`). Czy to wystarcza, by program działał prawidłowo?

d) Do klasy `Licznik` dodać dwie nowe metody: `void inkrementuj()` oraz `void dekrementuj()` , które modyfikują wartość licznika odpowiednio instrukcją `wartosc++` oraz `wartosc--` . Czy zastosowanie tych instrukcji spowodowało poprawne działanie programu?

e) Zwiększyć liczbę wykonawców do 8, przy czym wykonawcy o parzystych numerach dodają 1 do licznika, a wykonawcy o nieparzystych numerach odejmują -1. Jak zachowuje się program dla różnych wartości n w stosunku do wersji z 2 wątkami?

f) Zastosować słowo kluczowe `synchronized` do synchronizacji dostępu do wartości licznika, czy poprawiło to działanie programu?

g) Zmierzyć jak szybko działa program z synchronizacją w stosunku do wersji bez synchronizacji. Do pomiaru czasu można zastosować metodę `System.currentTimeMillis()` .

3. Napisać program symulujący system, w którym oprócz wątku głównego uruchamiany jest wątek roboczy – wykonujący długotrwałe obliczenia. Wątek główny uruchamia wątek obliczeniowy, a następnie na bieżąco, tzn. bez przerywania obliczeń, wyświetla na ekranie (co 0,1 sek.) komunikat o postępie obliczeń (w procentach), aż do zakończenia obliczeń w wątku roboczym.

   Wątek obliczeniowy implementowany jest przez klasę `Obliczenia` :

```
class Obliczenia implements Runnable {

    public Obliczenia(int ile) { }

    @Override
    public void run() { }

    public long dajWynik() { }

    /* Zwraca postęp obliczeń jako liczbę od 0.0 do 1.0 */
    public double dajPostep() { }

    /* Zwraca true jeżeli obliczenia zostały zakończone / przerwane */
    public boolean czyKoniec() { }

    private long wynik_ = 0;
    private final int ile_;
    ...
}
```

   Obliczenia wykonywane w metodzie `run()` polegają na dodawaniu 1 do pola `wynik_` tyle razy ile wynosi wartość zmiennej `ile` .

   - Aby obliczenia trwały dostatecznie długo należy odpowiednio dobrać wartość `ile` , np. 10^9.

- Dodać opcję przerywania obliczeń za pomocą metody `Thread.interrupt()` wywoływanej w wątku głównym po przekroczeniu 50% postępu obliczeń. W wątku obliczeniowym wymaga to użycia kodu:

  ```
  if (Thread.currentThread().isInterrupted()) {
      ...
  }
  ```

  Przykładowy wydruk:

  ```
  Postęp: 0,0 %
  Postęp: 3,8 %
  Postęp: 7,2 %
  Postęp: 9,6 %
  Postęp: 12,4 %
  Postęp: 16,0 %
  Postęp: 19,6 %
  Postęp: 22,5 %
  Postęp: 25,9 %
  Postęp: 29,4 %
  Postęp: 32,7 %
  Postęp: 36,3 %
  Postęp: 39,9 %
  Postęp: 43,5 %
  Postęp: 47,0 %
  Postęp: 50,7 %
  Obliczenia przerwane
  Wynik: 506746571
  ```

4. Napisać klasę `Sorter` sortującą tablicę liczb całkowitych w następujący sposób:

   - jeżeli rozmiar tablicy jest mniejszy od wartości progowej, to tablica sortowana jest np. za pomocą algorytmu sortowania przez proste wybieranie
   - jeżeli rozmiar tablicy jest większy od wartości progowej, to jest ona dzielona na dwie części, które są sortowane w osobnych wątkach przez nowoutworzone obiekty klasy `Sorter`, a następnie tak otrzymane posortowane podtablice scalane są w tablicę wynikową.