ZPO – Laboratorium 3

Zadanie 1 – komunikacja między wątkami

Utworzyć aplikację wielowątkową, wyświetlająca liczby od 1 do 100 w kolejności rosnącej.

Wątek nr 1:

Wyświetlanie liczb nieparzystych

Wątek nr 2:

Wyświetlanie liczb parzystych

Watki czekają na siebie wzajemnie tak aby wyświetlić naprzemiennie liczby nieparzyste i parzyste.

Przykładowe rozwiązanie:
Stworzyć dwie klasy „WatekParzyste", „WatekNieparzyste" dziedziczące po klasie Thread. Przesłonić metodę run, w której znajdzie się inkrementacja i wyświetlanie licznika.
W konstruktorach obu klasy przekazać jeden wspólny obiekt (Object) dzięki, któremu uda się zsynchronizować oba wątki. Ponieważ obiekt jest współdzielony użyj bloku synchronized, aby operacje na obiekcie były thread-safe. Wykorzystaj metody: notify() oraz wait(), do komunikacji między wątkami.

Przydatne linki:
https://docs.oracle.com/javase/tutorial/essential/concurrency/guardmeth.html
https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html
https://docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html
http://www.programcreek.com/2009/02/notify-and-wait-example/

Zadanie 2 – Typy generyczne

Stworzyć klasę generyczną (ClassName<T>), gdzie T jest dowolny typem obiektowym.

1. Klasa zawiera konstruktor z jednym parametrem typu T.
2. Przesłoń metodę toString tak aby zwrócić nazwę klasy, dla której została utworzona instancja obiektu (…getClass().getName())
3. Przykładowe użycie

```
KlasaGeneryczna<Integer> i = new KlasaGeneryczna<>(5);
KlasaGeneryczna<Double> d = new KlasaGeneryczna<>(5.0);
List<KlasaGeneryczna<?>> lista = new ArrayList<>();
lista.add(i);
lista.add(d);

System.out.println(lista);
```

4. Sprawdź i przetesuj działanie klas generycznych

https://docs.oracle.com/javase/tutorial/java/generics/
https://www.tutorialspoint.com/java/java_generics.htm