Zaawansowane Technologie Programistyczne
Laboratorium 1 – Wątki w języku Java.

**Zadanie 1.** [3p]

Utwórz tablicę liczb całkowitych o rozmiarze co najmniej 100000. Napisz program, który uruchamia 10 wątków – generatorów oraz jeden wątek – licznik. Wszystkie wątki mają mieć dostęp do tej samej tablicy. Każdy generator ma przydzielony fragment tablicy, który wypełnia losowymi liczbami (z zadanego zakresu). Po zapełnieniu całej tablicy, gdy wszystkie wątki generatora zakończą swoje działanie, ma zostać uruchomiony wątek licznika, który obliczy sumę i średnią elementów tablicy.

**Zadanie 2.** [2p]

Zasymuluj zjawisko Race Condition. Utwórz klasę **SharedVar**, która ma jedno pole **value** typu **long** i metody **inc()** oraz **dec()**, które odpowiednio zwiększają i zmniejszają wartość zmiennej **value.**
Napisz program, w którym pewna liczba wątków (np. 10) współdzieli tą samą instancję **SharedVar**. Każdy wątek określoną liczbę razy (np. 1000000) będzie zwiększać wartość zmiennej dzielonej. Po zakończeniu działania wątków należy wypisać wartość zmiennej value. Sprawdzić, jak na wynik końcowy wpływa użycie synchronizacji.

Sprawdzić, jak zachowuje się program bez synchronizacji, ale z wykorzystaniem java.util.concurrent.AtomicLong

**Zadanie 3.**

Napisz program symulujący problem producent-konsument.

Parametry: N – rozmiar bufora, P – liczba producentów, K – liczba konsumentów. Kod symulujący działanie producentów i konsumentów powinien być uruchomiony w K+P wątkach. Synchronizacja producentów i konsumentów ma być realizowana przez:

a. wait, notifyAll, synchronized [2p]
b. zamki implementujące interfejs Lock [2p]
c. kolejkę blokującą [1p]

**UWAGA**: W podpunktach a i b bufor cykliczny ma być zrealizowany w oparciu o **tablicę** N-elementową.

**Wskazówka**: Utworzyć interfejs lub klasę abstrakcyjną Bufor, w każdym podpunkcie zadania dostarczyć inną implementację bufora.

Pomoc:

http://docs.oracle.com/javase/tutorial/essential/concurrency

http://docs.oracle.com/javase/7/docs/api