

## Programowanie funkcyjne i współbieżne

### Lista 9

1. Przeanalizuj poniższy program. Dwa wątki zwiększają 200 000 razy wspólny licznik o 1. Po ich zakończeniu wartość licznika powinna oczywiście wynosić 400 000. Uruchom ten program kilka razy.

```
object Zad1 extends App {
  var counter = 0          // counter variable

  def readWriteCounter: Unit = {
    val incrementedCounter = counter + 1 // reading counter
    counter = incrementedCounter         // writing to counter
    // counter += 1                       // shorter code
  }

  val p = new Thread() => for(_ <- 0 until 200000) readWriteCounter
  val q = new Thread() => for(_ <- 0 until 200000) readWriteCounter
  val startTime = System.nanoTime
  p.start; q.start
  p.join; q.join
  val estimatedTime = (System.nanoTime - startTime)/1000000
  println(s"The value of counter = $counter")
  println(s"Estimated time = ${estimatedTime}ms, Available processors = ${Runtime.getRuntime.availableProcessors}")
}
```

- Jak wyjaśnisz różne wartości licznika? W wyjaśnieniu załóż, że mamy tylko jeden procesor, więc nie ma akcji jednoczesnych. Wskaż fragment kodu, który jest źródłem problemów. Odpowiedź należy umieścić w postaci komentarza na początku pliku Lista9.scala.
  - Popraw powyższy program, wykorzystując mechanizm kodu synchronizowanego (blokada wewnętrzna, monitory).
  - Popraw powyższy program, wykorzystując mechanizm semaforów (klasa `java.util.concurrent.Semaphore`).
2. Zaimplementuj metodę `parallel`, która jako argumenty bierze dwa bloki kodu, wykonuje je jednocześnie w osobnych wątkach i zwraca wyniki ich obliczeń w postaci pary.
- ```
def parallel[A, B](block1: =>A, block2: =>B): (A, B) = ???
```

Przykładowe testy:

```
println(parallel("a"+1, "b"+2))
println(parallel(Thread.currentThread.getName, Thread.currentThread.getName))
```

3. Zaimplementuj metodę `periodically`, która jako argumenty bierze interwał czasowy `duration` (w milisekundach), maksymalną liczbę powtórzeń `times` oraz blok kodu `block`. Metoda tworzy wątek demona, który wykonuje podany blok kodu z pauzami trwającymi `duration` milisekund, maksymalnie `times` razy.

```
def periodically(duration: Long, times: Int)(block: => Unit): Unit = ???
```

Test (na końcu metody `main`) może wyglądać tak:

```
periodically(1000, 5){print("y ")}
periodically(1000, 25){print("x ")}
Thread.sleep(10000)
println("Done sleeping")
```

Uruchom ten program kilka razy. Wynik powinien być taki (z dokładnością do przepłotu):

```
y x y x x y x y x y x x x x Done sleeping
```

Dlaczego "x" nie zostało wyświetlone 25 razy? (Odpowiedź umieść w komentarzu.)

Każde zadanie (Zad1Mon, Zad1Sem, Zad2, Zad3) ma być niezależną aplikacją z testami w metodzie `main`. Wszystkie zadania należy umieścić w pliku `Lista9.scala`.

UWAGA. Proszę nie używać arkusza elektronicznego Scali w IntelliJ. Programy współbieżne nie będą wykonywane poprawnie!