# Programowanie funkcyjne i współbieżne
## Lista 10

1. Na wykładzie 9 (str. 42-43) był przedstawiony program producent/konsument z ograniczonym buforem cyklicznym.
   a) Przepisz ten program, wykorzystując zamiast klasy BoundedBuffer klasę biblioteczną java.util.concurrent.ArrayBlockingQueue.
   b) W programie z podpunktu a) utwórz kilka producentów i konsumentów. Nadaj im unikatowe nazwy, np. Producer1, Consumer1 itd. W jednym z testów utwórz dwa producenty i trzy konsumenty. Dlaczego program się nie kończy? Odpowiedź umieść w komentarzu.
   c) Z programu w podpunkcie b) usuń definicje klas Producer i Consumer. Wykorzystaj ExecutionContext do wykonywania odpowiadających im zadań. W jednym z testów utwórz dwa producenty i trzy konsumenty. Dlaczego program się kończy? Odpowiedź umieść w komentarzu.

2. Napisz program, rozwiązujący problem ucztujących filozofów (wykład 9, str. 38) dla $N$ filozofów za pomocą semaforów (java.util.concurrent.Semaphore). Rozwiązanie powinno spełniać następujące warunki:

1. Każdy filozof ma stałe miejsce przy stole. Filozof je tylko wtedy, gdy ma dwie pałeczki.
2. Dwóch filozofów nie może jednocześnie trzymać tej samej pałeczki.
3. Nie występuje blokada (sytuacja patowa). Może ona wystąpić np. wtedy, gdy wszyscy filozofowie podniosą lewe pałeczki i będą czekać na zwolnienie prawych.
4. Nikt nie może być zagłodzony. Oczywista z pozoru strategia, polegająca na poczekaniu, aż obie pałeczki będą wolne, może spowodować zagłodzenie dwóch filozofów (dlaczego?).
5. Żaden z filozofów nie zajmuje się tylko jedzeniem. Po zakończeniu posiłku każdy odkłada pałeczki i wraca do sali medytacji.
6. Filozofowie podnoszą i odkładają pałeczki po jednej naraz.
7. Nie można wyróżniać żadnego z filozofów (algorytmy ich działania powinny być takie same).

Jedno z rozwiązań zakłada, że na początku wszyscy filozofowie medytują w przeznaczonej do tego sali, natomiast posiłki spożywają w jadalni. Należy zaangażować odźwiernego, pilnującego drzwi jadalni i pozwalającego przebywać w niej jednocześnie co najwyżej $N$-1 filozofom. Dzięki temu co najmniej dwom filozofom, siedzącym przy stole, brakuje co najmniej jednego sąsiada, a zatem co najmniej jeden filozof może jeść (dlaczego?). Każdy filozof ma wyświetlać odpowiednie komunikaty, informujące o: czasie medytacji, wejściu do jadalni, czasie jedzenia, wyjściu z jadalni.

Wskazówka. Przedstaw filozofów jako wątki (każdy filozof w pętli naprzemiennie medytuje i posila się), sekcją krytyczną jest jedzenie, a zasobami dzielonymi są pałeczki do ryżu. Wątki są ponumerowane od 0 do $N$-1, co odpowiada stałym miejscom filozofów przy stole i wykonują się współbieżnie. Użycie każdej pałeczki jest kontrolowane przez semafor binarny, a odźwierny jest reprezentowany przez semafor ogólny z wartością początkową $N$-1.

Każde zadanie (Zad1a, Zad1b, Zad1c, Zad2) ma być niezależną aplikacją z testami w metodzie main. Wszystkie zadania należy umieścić w pliku Lista10.scala.