Model checking programów JAVA w Java Pathfinder

1. Wstęp

W każdym zadaniu należy napisać program JAVA bazujący na tych samych plikach startowych i uruchmić go poprzez JPF. Należy przedstawić końcowy program wraz z czasem i zużyciem pamięci przez JPF podczas weryfikacji plików. Zużycie czasu i pamięci należy przedstawić w następującej tabelce:

Specyfikacja CPU = XYZ GHz PQR Cores Pamięć = XYZ MB

Key	Czas (w sekundach)	Pamięć (MB)
n1	t1	m1
n2	t2	m2
n3	t3	m3
⋮	⋮	⋮

Pozycja *Key* będzie doprecyzowana w każdym z zadań. Należy zwiększać tę wartość dopóki czas wykonywania eksperymentu nie przekroczy 900 sekund (co należy oznaczyć gwiazdką w rubryce czasu).

2. Problem biesiadujących filozofów

Problem biesiadujących filozofów (DP) to jeden z klasycznych problemów z "zacięciem", o którym więcej można przeczytać chociażby tutaj:

`http://en.wikipedia.org/wiki/Dining_philosophers_problem`

Istnieje szereg sposobów na uniknięcie zacięcia. Naszym zadaniem będzie wymodelowanie i zweryfikowanie dwóch z możliwych podejść do problemu. Generyczny sposób uruchamiania każdego z programów poświęconych DP poprzez JPF to:

jpf <DP nazwa klasy><liczba filozofów>

2.1. **Zadanie 1: Problem biesiadujących filozofów z zacięciem.**

Plik startowy DP.java:

`http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/DP.java`

zawiera wersję biesiadujących filozofów z zacięciem. Należy uruchomić ten program poprzez JPF i przekonać się, że faktycznie się "zacina". Następnie należy uzupełnić tabelkę z wynikami pomiarów stosując liczbę filozofów jako *Key*.

2.2. **Zadanie 2: Problem filozofów z jednym leworęcznym filozofem.**

Pozbędziemy się zacięcia zmuszając jednego z filozofów do zabrania widelca w innej kolejności od pozostałych. Plik DPInnaKolejnosc.java:

`http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/DPInnaKolejnosc.java`

zawiera podstawowy pomysł na to, jak to zrobić. W obecnej wersji plik jest taki sam, jak DP.java. Należy poprawić go tak, aby osiągnąć pożądany rezultat. Nie wolno dodawać żadnych dodatkowych metod, pól lub klas, ale po prostu zmodyfikować istniejącą metodę. Następnie należy uzupełnić tabelkę z wynikami pomiarów stosując liczbę filozofów jako *Key*.

2.3. **Zadanie 3: Problem filozofów z kelnerem.**

Pozbędziemy się teraz zacięcia wykorzystując kelnera, który pozwoli $k-1$ filozofom opuścić stolik, gdzie k jest całkowitą liczbą filozofów. Plik DPKelner.java:

`http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/DPKelner.java`

zawiera podstawowy pomysł na to, jak to zrobić. W obecnej wersji plik zawiera błąd, który spowoduje, że zacięcie nie zostanie wyeliminowane. Należy poprawić ten błąd i uruchomić JPF celem przetestowania nowej wersji programu. Nie wolno dodawać żadnych dodatkowych metod, pól lub klas, ale po prostu

zmodyfikować istniejącą metodę. Następnie należy uzupełnić tabelkę z wynikami pomiarów stosując liczbę filozofów jako *Key*.