Teoria współbieżności
III Rok

Informatyka
Wydział Informatyki, Elektroniki i Telekomunikacji

Problem pięciu filozofów

laboratorium nr 3

Kamil Koczera

Semestr zimowy

2020/2021

Spis treści

1 Wstęp

1.1 Informacje o archiwum

W archiwum znajdują się dwa foldery: java oraz js, zawierające rozwiązania odpowiednio w javie oraz javacript. W każdym z tych folderów znajdują się foldery: plots (zawierający wygenerowane wykresy) oraz data (zawierający pliki z otrzymanymi czasami oczekiwania). Dla każdego przypadku niepowodującego zakleszczenia wykonałem program z różnymi parametrami: 5, 9 i 15 filozofów, każdy jedzący 10, 50 lub 100 posiłków. Do wygenerowania wykresów został wykorzystany średni (rzeczywisty) czas oczekiwania na rozpoczęcie jedzenia. Założyłem, że "jedzenie" trwa 10 ms.

1.2 Wersja java

Obydwa rozwiązania (z zagłodzeniem i z arbitrem) znajdują się w jednym projekcie. Klasy "Philosohper", "Main" (z metodą "main") oraz "Fork" dotyczą rozwiązania z zagłodzeniem, natomiast klasy "PhilosopherArbiter", "MainArbiter" (z metodą "main") oraz "ForkArbiter" dotyczą, jak sama nazwa wskazuje, rozwiązania z arbitrem. Wykonanie programów wyświetli przebieg operacji (rozpoczynanie jedzenia, kończenie), średnie czasy oczekiwania oraz zapisze je do plików.

1.3 Wersja javascript

Wszystkie rozwiązania (naiwne, asynchroniczne, z arbitrem, zagłodzeniem) znajdują się w pliku "phil5.js". Rozwiązania te opierają się na funkcji "setTimeout()". Jako drugi parametr przyjmuje ona czas, po którym dodaje callback do stosu ramek czekających na wykonanie. Jeśli jest zainstalowany node.js to można wykonać program z przykładowymi wartościami poleceniem:

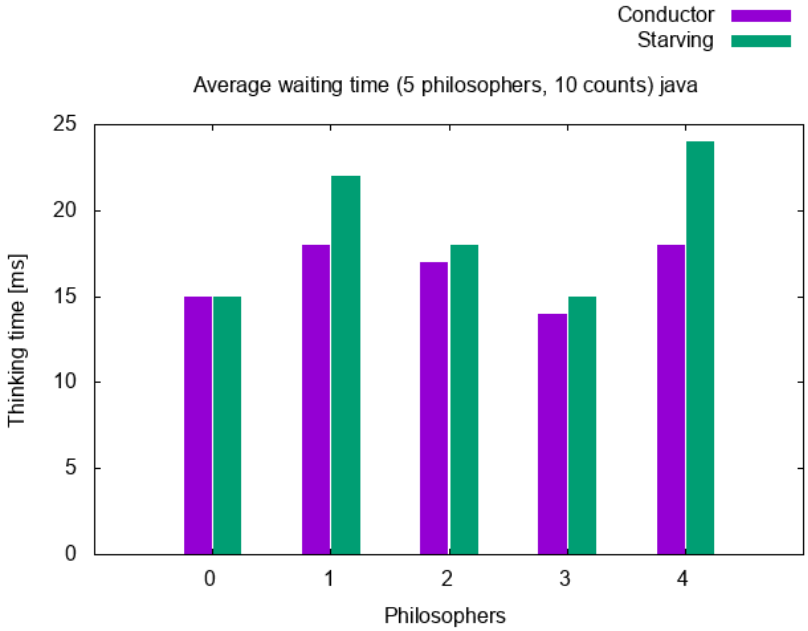
```
$ node phil5.js
```

Program wykona trzy sposoby rozwiązania problemu pięciu filozofów (pominie rozwiązanie naiwne ze względu możliwości zakleszczenia), podobnie jak w wersji "javowej" wyświetli przebieg operacji (rozpoczynanie jedzenia, kończenie), średnie czasy oczekiwania oraz zapisze je do plików.

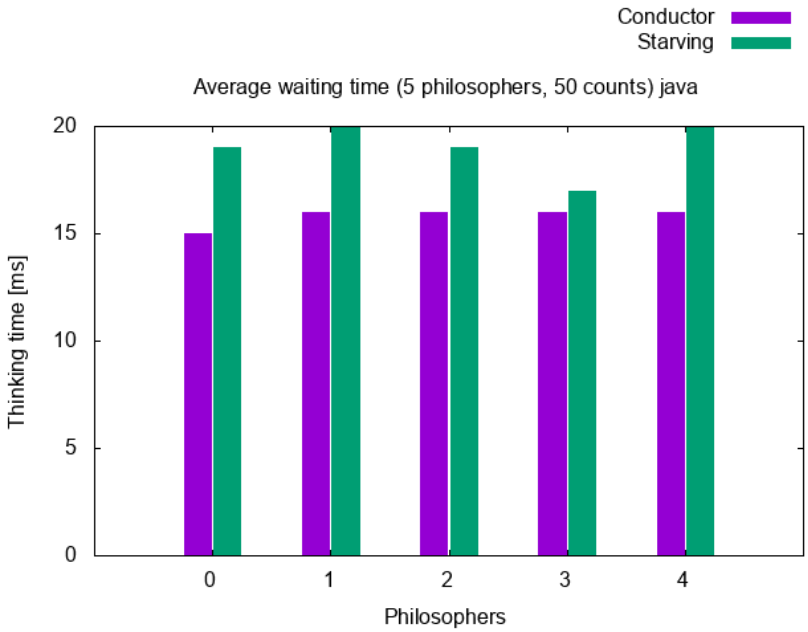
2 Wykresy

Wykresy jakie otrzymałem na podstawie przeprowadzonych przeze mnie testów prezentują się następująco:

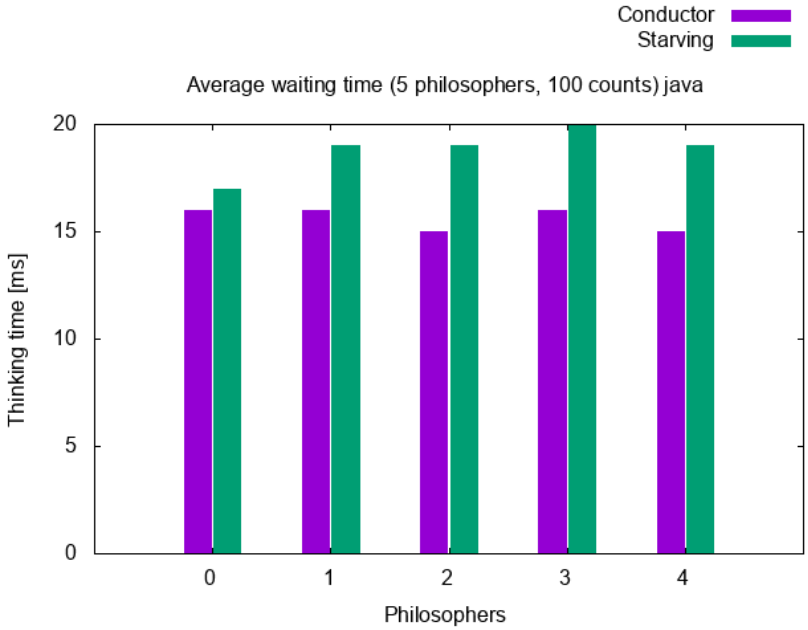
2.1 Java



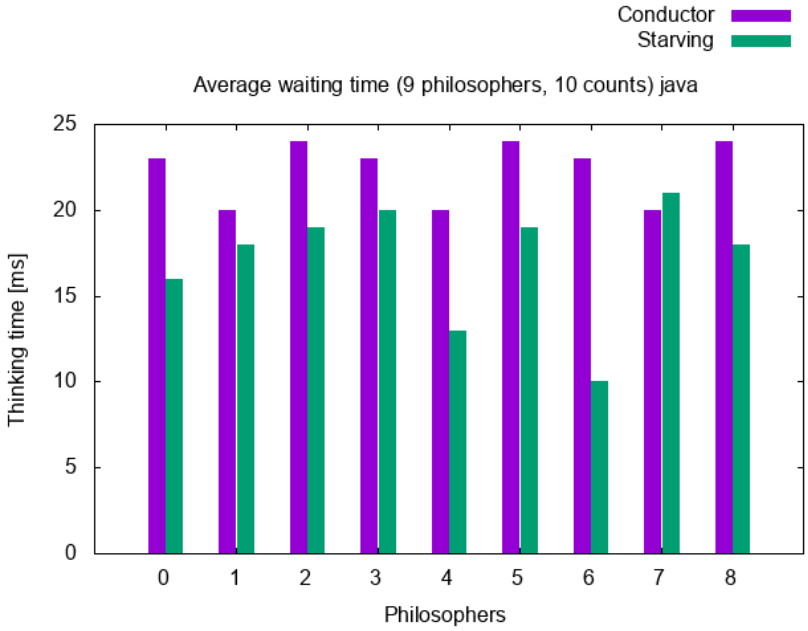
Rysunek 1: 5 filozofów, każdy mający do spożycia 10 dań



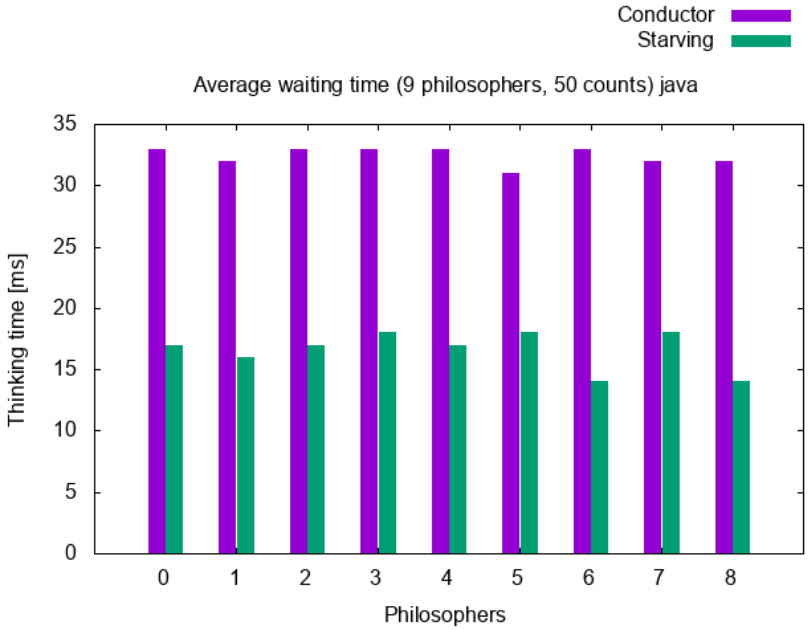
Rysunek 2: 5 filozofów, każdy mający do spożycia 50 dań



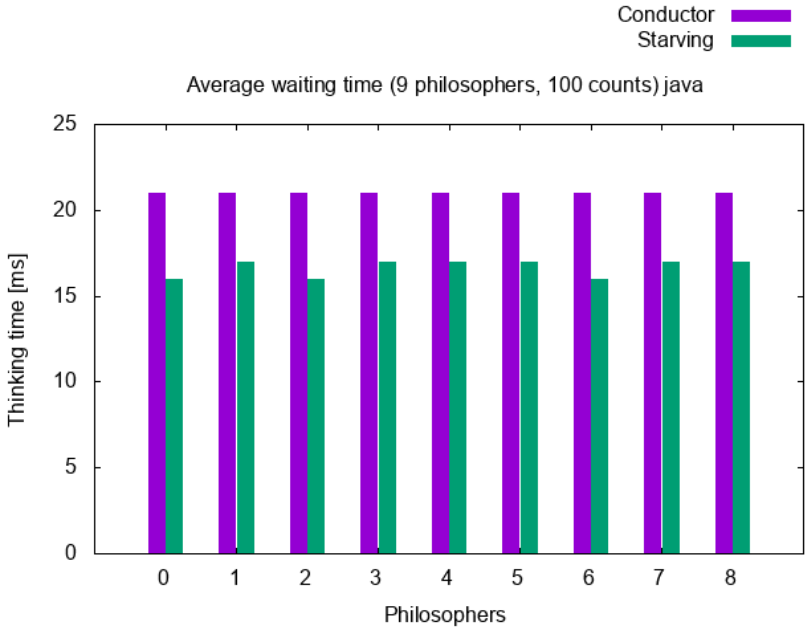
Rysunek 3: 5 filozofów, każdy mający do spożycia 100 dań



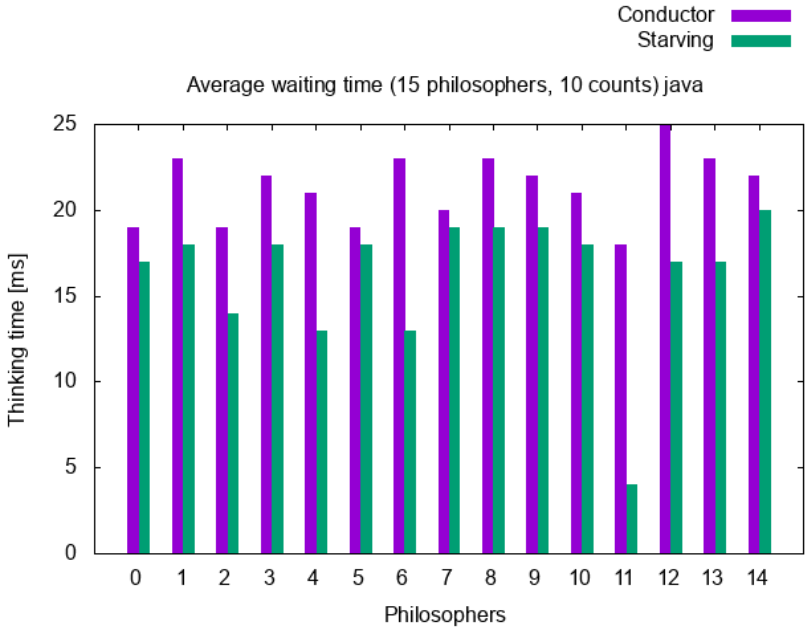
Rysunek 4: 9 filozofów, każdy mający do spożycia 10 dań



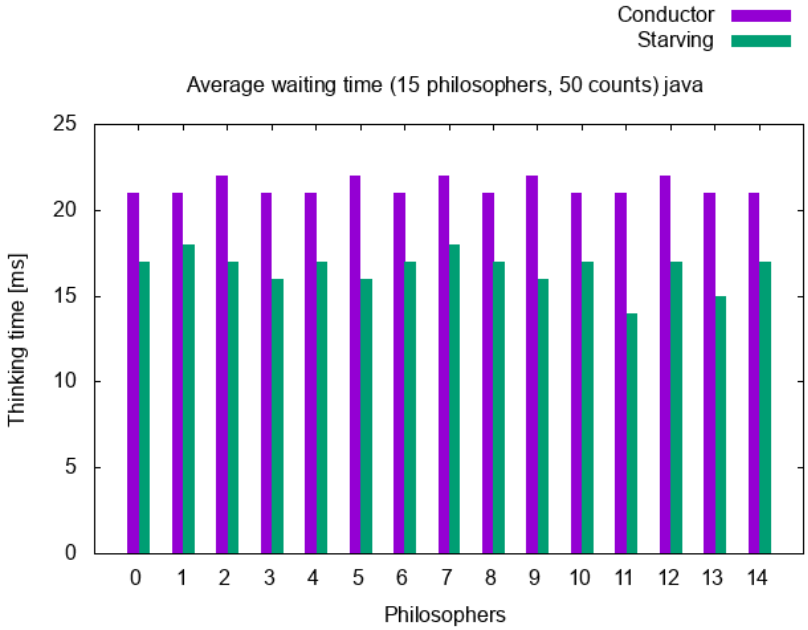
Rysunek 5: 9 filozofów, każdy mający do spożycia 50 dań



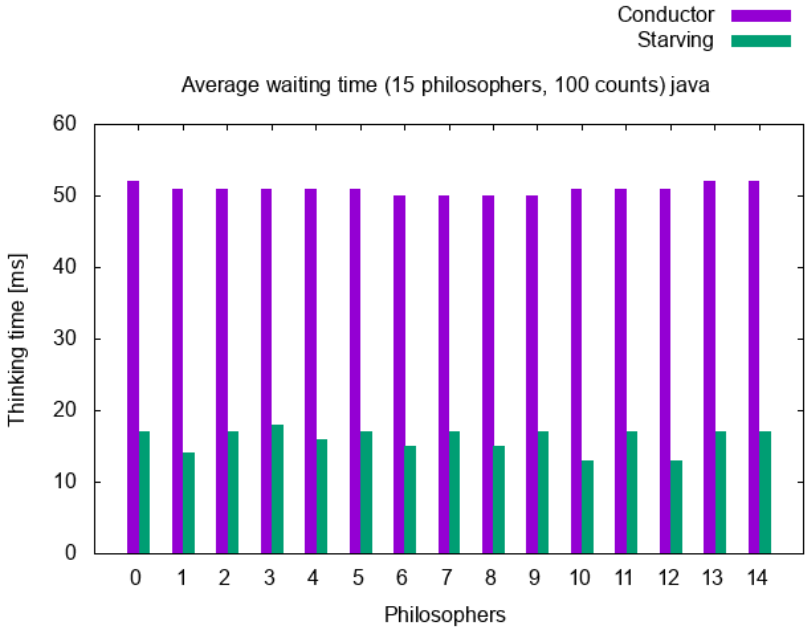
Rysunek 6: 9 filozofów, każdy mający do spożycia 100 dań



Rysunek 7: 15 filozofów, każdy mający do spożycia 10 dań

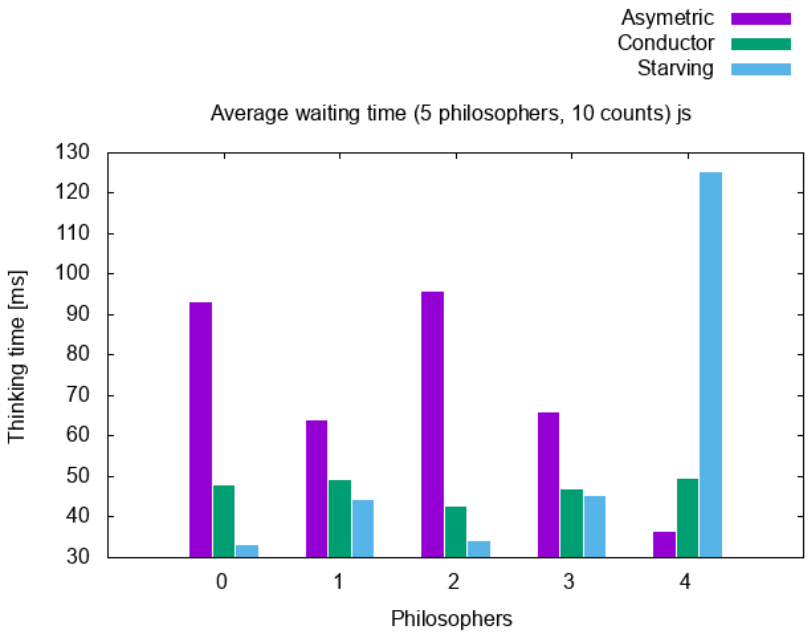


Rysunek 8: 15 filozofów, każdy mający do spożycia 50 dań

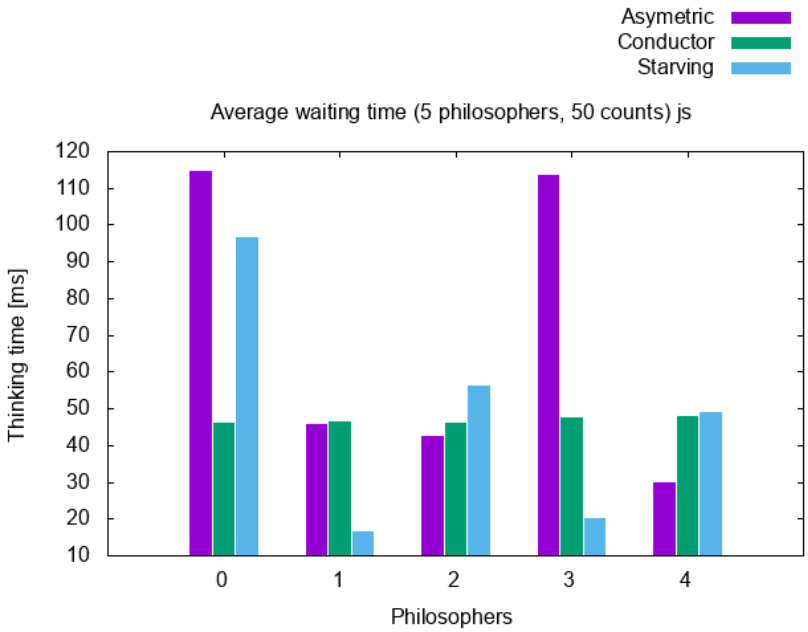


Rysunek 9: 15 filozofów, każdy mający do spożycia 100 dań

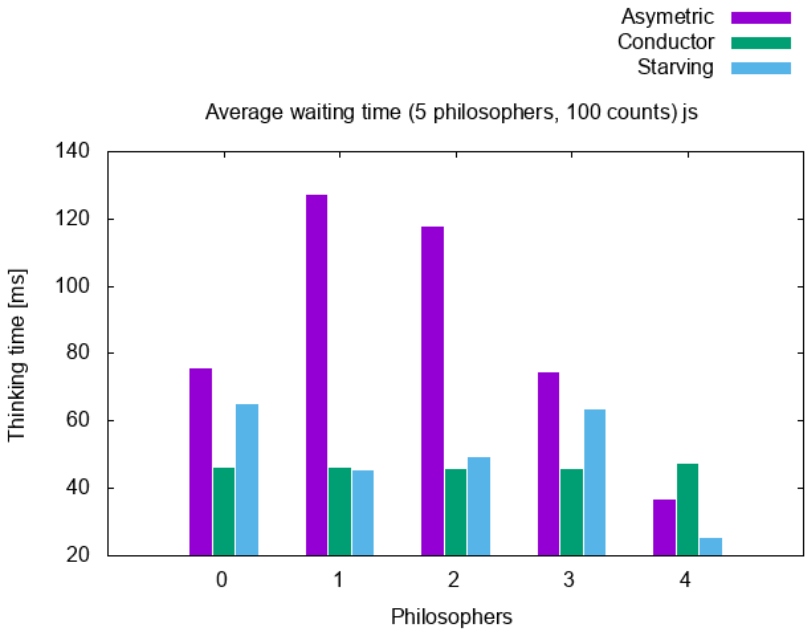
2.2 Javascript



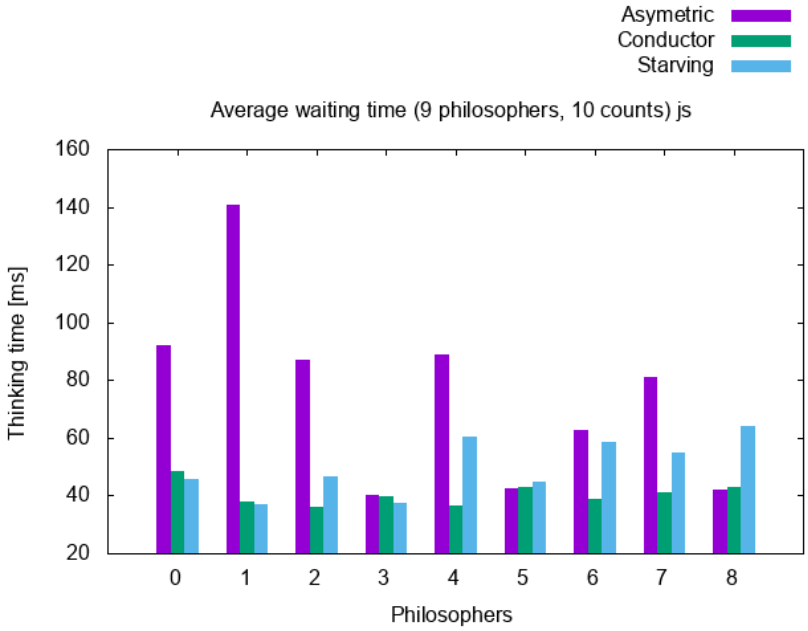
Rysunek 10: 5 filozofów, każdy mający do spożycia 10 dań



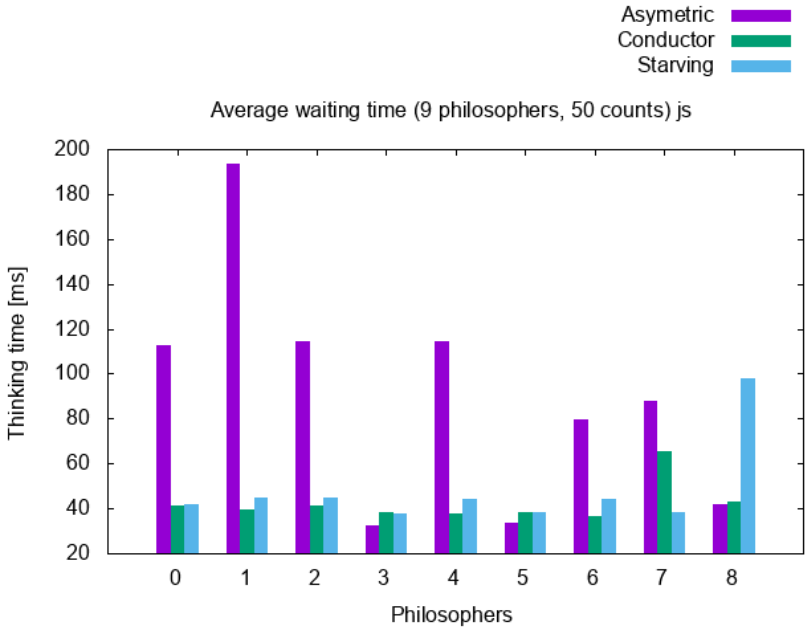
Rysunek 11: 5 filozofów, każdy mający do spożycia 50 dań



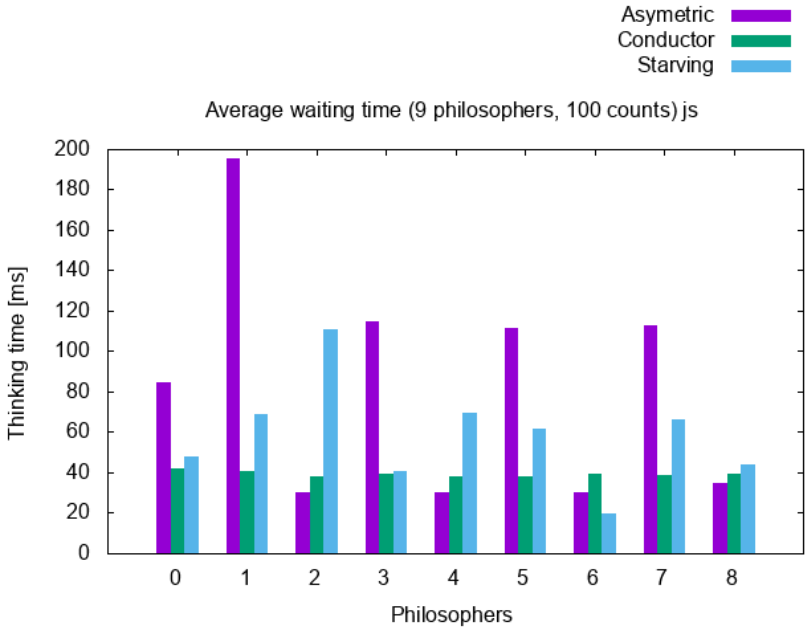
Rysunek 12: 5 filozofów, każdy mający do spożycia 100 dań



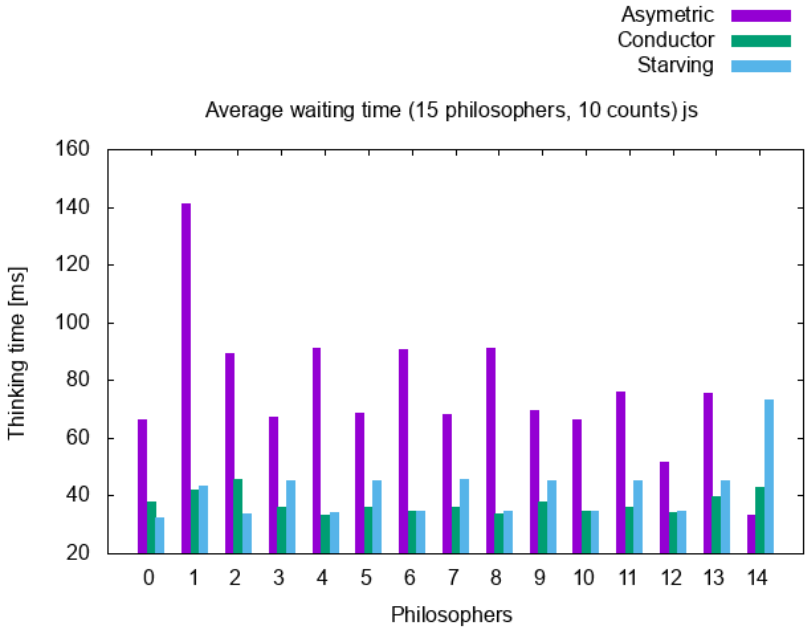
Rysunek 13: 9 filozofów, każdy mający do spożycia 10 dań



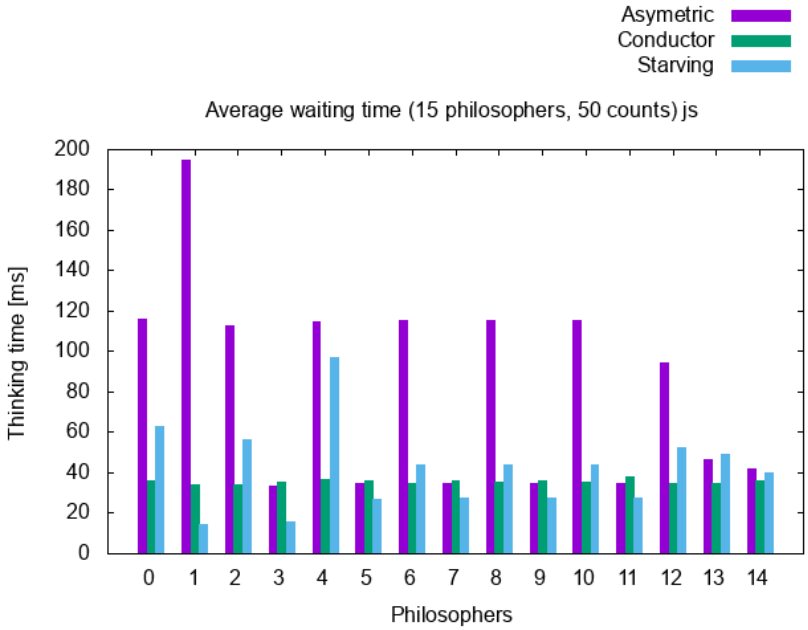
Rysunek 14: 9 filozofów, każdy mający do spożycia 50 dań



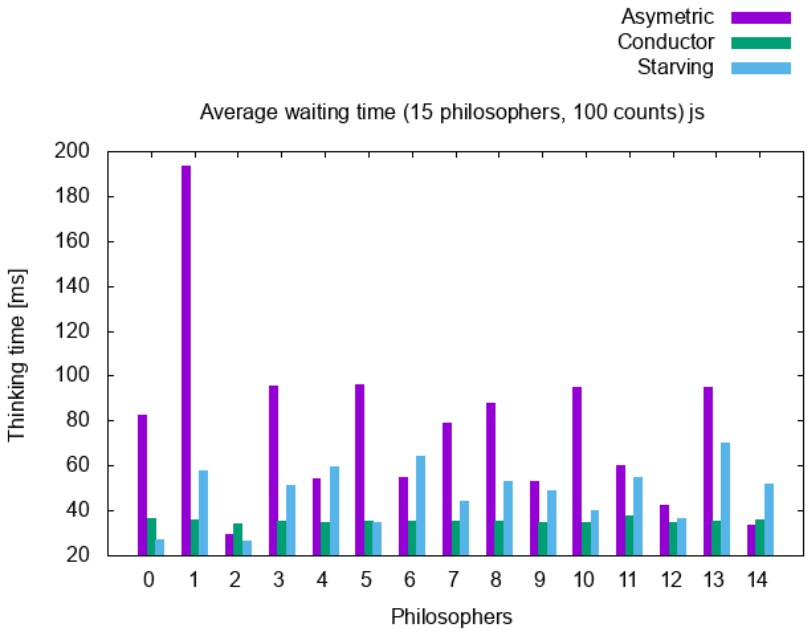
Rysunek 15: 9 filozofów, każdy mający do spożycia 100 dań



Rysunek 16: 15 filozofów, każdy mający do spożycia 10 dań



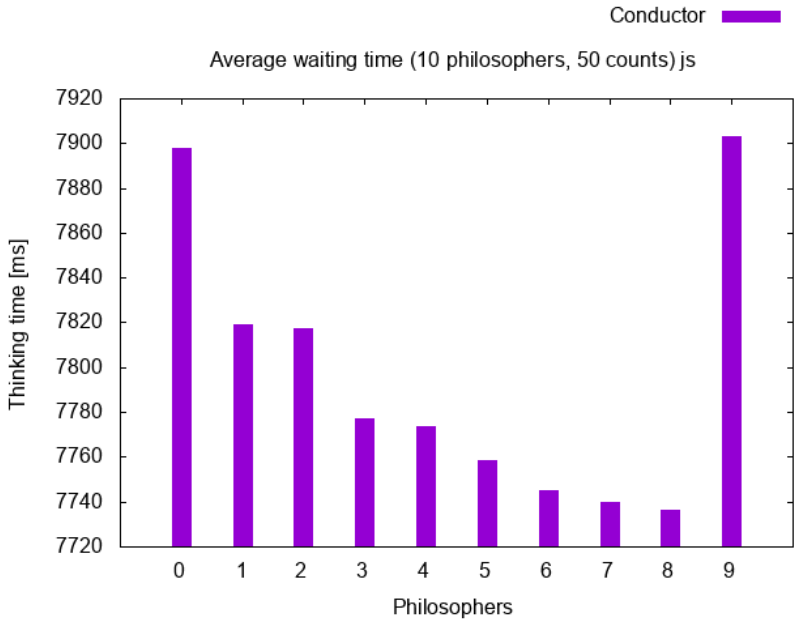
Rysunek 17: 15 filozofów, każdy mający do spożycia 50 dań



Rysunek 18: 15 filozofów, każdy mający do spożycia 100 dań

3 Wnioski

Na podstawie implementacji javowej wyraźnie widać, że przy niewielkiej ilości filozofów czasy oczekiwania obu metod są zbliżone, z lekką przewagą dla arbitra. Natomiast wraz ze wzrostem liczby filozóf rośnie czas oczekiwania w metodzie kelnera (kelner ma więcej "klientów" do obsłużenia). Jednak zarówno w przypadku implementacji javowej i js-owej można zauważyć, że metoda z kelnerem sprawiedliwiej traktuje filozofów (każdy ma bardzo podobny czas oczekiwania, natomiast w pozostałych przypadkach te czasy zazwyczaj się różnią). W implementacji js-owej zauważyć można, że metoda asymetryczna często daje taki efekt, że co drugi filozof ma wysoki czas oczekiwania, a co drugi niski. Jest to zapewne efekt użtego środowiska oraz sposób w jaki działa wspomniana we wstępie funkcja "setTimeout()". Metoda z zagłodzeniem daje lepsze wyniki od metody asymetrycznej. Wynika to z faktu, że choć w teorii metoda ta jest najbardziej podatna na zagłodzenie, to w praktyce szansa na to, że filozofowi nie uda się podnieść widelców wiele razy pod rząd jest minimalna. Najbardziej optymalnym rozwiązaniem w implementacjach js-owych wydaje się być wersja z kelnerem. Jednak użyłem tutaj zawłaszczania dwóch widelców naraz przez filozofów, ze względu na fakt, że podczas korzystania z rozwiązania najpierw jeden widelec, a później drugi program działał prawidłowo, ale z bardzo dużymi czasami oczekiwania (przeprowadzenie porównania na wykresach w takiej sytuacji było niemożliwe). Testowałem to rozwiązanie kilkukrotnie przy różnych parametrach i efekt był ten sam. Poniżej zamieszczam wykres, wykorzystujący ten wariant przejmowania widelców dla 10 filozofów i 50 posiłków (proszę zwrócić uwagę na wartości na osi y). Przeprowadzenie porównania między wersjami javowymi, a js-owymi jest trudne do przeprowadzenia, m. in. ze względu na fakt, że wersja javowa może działać na kilku wątkach jednocześnie, synchronizując jedynie pracę wątków w poszczególnych momentach, podczas gdy w js-ie wszytkie operacje odbywają się w ramach jednego wątku, w którym po kolei wykonywane są wywołania funkcji czekające w kolejce.



Rysunek 19: 10 filozofów, każdy mający do spożycia 50 dań (wersja z arbitrem - najpierw jeden widelec, później drugi)