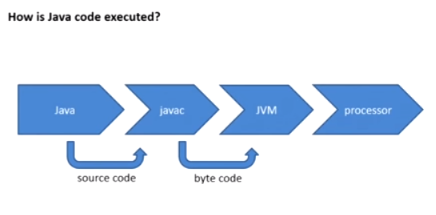
**Java Memory Model**

**Czym jest Java Memory Model?**

Napisany kod źródłowy jest ciągiem instrukcji, które pełnią rolę wskazówek dla wirtualnej maszyny Javy co ma zostać wykonane. Kod źródłowy jest kompilowany przez kompilator do kodu bajtowego i w takiej postaci trafia do wirtualnej maszyny Javy, która optymalizuje nasz kod, przekształcając go zgodnie z pewnymi zasadami. Java Memory Model jest właśnie modelem, który nakłada na JVM ograniczenia dotyczące tych przekształceń.

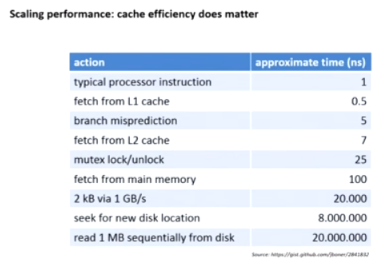
Odpowiada za to, że odczyt jakiejś wartości z pamięci zwróci oczekiwaną wartość.



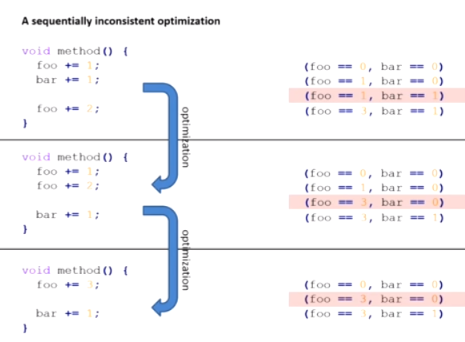
Wykonanie jakiejkolwiek instrukcji procesora polega na wysłaniu tej instrukcji do procesora, który w uproszczeniu:

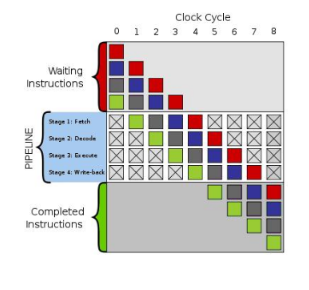
* pobiera instrukcję,
* dekoduje pobraną instrukcję oraz pobiera z pamięci potrzebne dane,
* wykonuje instrukcję,
* zapisuje z powrotem wygenerowany rezultat w pamięci,

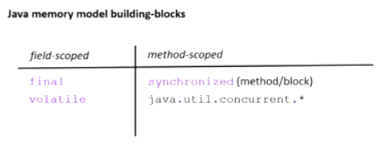
Najbardziej czasochłonnymi operacjami procesora są operacje zapisu i odczytu z pamięci dlatego procesor nigdy nie operuje bezpośrednio na pamięci, a posiada swoją własną pamięć cache, której używa w celu optymalizacji swojego działania.



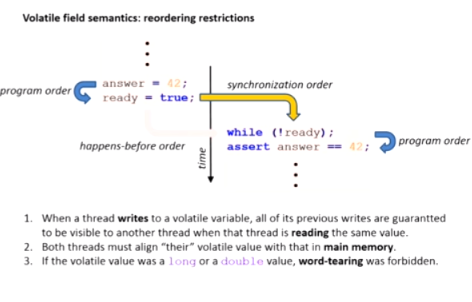
W przypadku jednowątkowych aplikacji optymalizacja jest stosunkowo prosta, ale problem pojawia się w przypadku programów wielowątkowych.





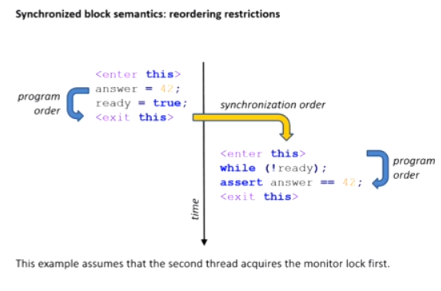


Słowo kluczowe **volatile** informuje JVM, że zmienna może być modyfikowana przez więcej niż jeden wątek. Oznaczenie zmiennej jako volatile nakłada na JVM ograniczenia co do przekształcenia kodu związanego z tą zmienną, nie pozwala na optymalizację przez zmiane kolejności wykonania instrukcji oraz nakłada na każdy z wątków obowiązek synchronizacji wartości zmiennej z główną pamięcią przy każdej operacji odczytu i zapisu jej wartości (a nie z cache).



**synchronized**

Podobnie jak volatile instruuje wirtualną maszynę, aby przy optymalizacji zachowała konkretną kolejność wykonania instrukcji oraz nakłada obowiązek synchronizacji z główną pamięcią.



**final**

Jeśli obiekt ma pole oznaczone jako final Java Memory model zapewnia, że ten obiekt zostanie w pełni zainicjowany zanim inny wątek dostanie do niego dostęp.