

## Wstęp

Java swoją przenośność i bezpieczeństwo zawdzięcza swojej maszynie wirtualnej. Jest ona konieczna do uruchomienia jakiejkolwiek aplikacji Javy.

Sytuacja wygląda tak samo w przypadku Javy ME. Każde urządzenie, na którym chcemy uruchomić aplikację Javy, musi mieć zaimplementowaną odpowiednią maszynę wirtualną.

Odpowiednią, bo zgodną ze specyfikacją konfiguracji, którą implementuje. Plus ewentualne rozszerzenia zgodne z profilem. Plus rozszerzenia dodawane przez producentów urządzeń.

Konfiguracja definiuje minimalną funkcjonalność swojej maszyny wirtualnej wymieniając te części standardowej maszyny wirtualnej, których ta nie musi posiadać.

W tym rozdziale przedstawię cechy maszyny wirtualnej javy, która opisana jest w specyfikacji CLDC.

Konfiguracja ta bardzo ogranicza funkcjonalność swojej maszyny wirtualnej. Jest to spowodowane ograniczeniami nałożonymi przez urządzenia przenośne (mała ilość pamięci, wolne procesory), dla których zaprojektowana jest ta konfiguracja.

Sun zaproponował swoją implementację maszyny wirtualnej opisaną w specyfikacji CLDC. Nazywa się ona KVM od Kilobyte Virtual Machine. Przyjęło się nazywać tak każdą maszynę wirtualną pracującą na urządzeniu implementującym profil oparty na CLDC.

## Różnice z JVM

Poniżej przedstawię główne cechy różniące standardową maszynę Javy od tej opisaną przez CLDC i jej profile:

1. Brak operacji na liczbach zmiennoprzecinkowych.

Jest to spowodowane tym, że procesory większości urządzeń tej klasy nie realizują takich działań, a ich programowa emulacja jest zbyt kosztowna. Implikuje to oczywiście brak zmiennych i stałych czy tablic typu `Float` czy `Double`,

2. Brak tych cech języka, które bądź były bardzo pamięciożerne, bądź też mogły zagrozić ingerencją aplikacji w system plików urządzenia i dostępem do poufnych informacji.

Brak więc tutaj takich funkcji języka, jak: JNI (pamięciożerny, jednocześnie niebezpieczny dla środowiska), finalizacja obiektów (zbyt duży koszt przy małych korzyściach), czy słabe referencje (również dla oszczędności czasu procesora podczas działania aplikacji),

3. Inaczej wygląda tutaj weryfikacja klas.

Jest dwuetapowa. Pierwszym etapem jest "preweryfikacja" (ang. *preverification*). Następuje ona zaraz po kompilacji kodu źródłowego i polega na najbardziej pamięciożernej analizie kodu bajtowego (wcześniej wykonywanej przy uruchomieniu). Etap ten wykonywany jest *przed* instalacją aplikacji w urządzeniu przenośnym. Jego wyniki zapisywane są w plikach `.class`.

Drugi etap wykonywany jest podczas ładowania klasy. KVM korzystając z informacji zapisanych podczas "preweryfikacji" oraz wykonując szybki przegląd byte-codu sprawdza, czy wszystkie reguły języka są spełnione.

Jak widzimy, na MIDlety nałożone są poważne ograniczenia. Jest to sytuacja bardzo podobna do appletów pobieranych z niepewnego źródła. Dlatego o MIDletach również mówi się, że są uruchamiane "w piaskownicy" (ang. *sandbox*).

W/w różnice są jedynie tymi najbardziej znaczącymi. Bardziej szczegółowy opis znajduje się np. na stronie <http://java.sun.com/products/cldc/wp/>.

### **KVMy innych producentów**

Każdy producent przenośnych urządzeń implementuje swoją własną KVM. Jej budowa jest przecież zależna od platformy systemowej urządzenia. Producenci dodają również własną funkcjonalność, jak np. kontrolę nad wibratorem czy podświetlaniem ekranu w telefonie, dostęp do części książki telefonicznej, możliwość wysyłania SMS-ów czy nawet wykonania telefonu!

Do własnej maszyny wirtualnej dostarczają oni oczywiście na swoich stronach jej API. Warto się więc zapoznać z możliwościami konkretnych urządzeń. Pisząc jednak aplikacje, musimy pamiętać, że użycie charakterystycznych dla urządzenia funkcji zawęzi grupę odbiorców naszej aplikacji nie tylko do posiadaczy telefonów danej marki, ale czasem nawet do konkretnej wersji oprogramowania zainstalowanego na urządzeniu!!