Java Virtual Machine

Wstęp – dlaczego tak?

Rozwój sieci i komplikacja urządzeń przyczyną wzrostu popularności Javy.

- Tworzenie platform – duża przenośność
- Wzrost bezpieczeństwa – środowisko, zapobieganie
- Dystrybucja – małe porcje, aktualizacje

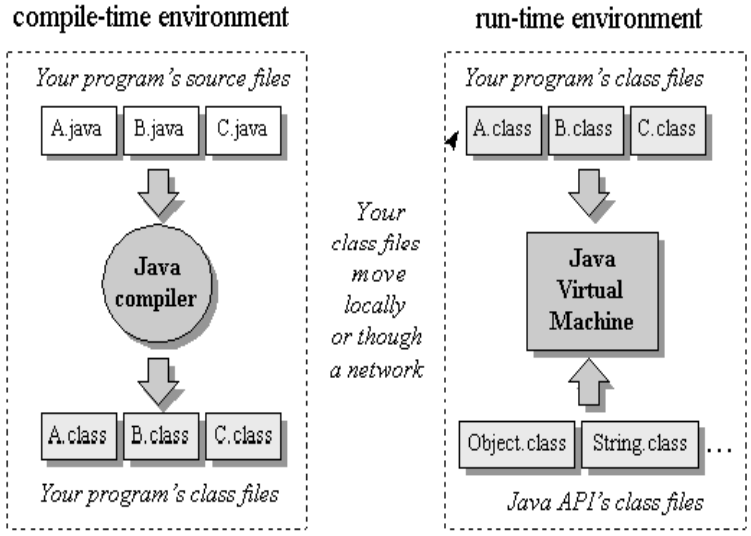
Tworzenie programu

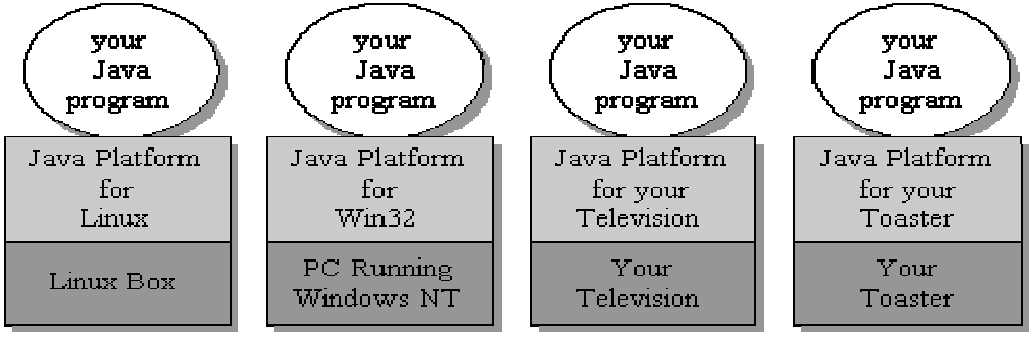
Kod w Javie -> kompilacja do .class

Pliki .class uruchamiamy na JVM

Java Application Programming Interface (API)

JVM + API = platforma





Co to jest JVM?

Wirtualna, bo jest to abstrakcyjny komputer zdefiniowany przez specyfikację.

Mówiąc JVM możemy myśleć o:

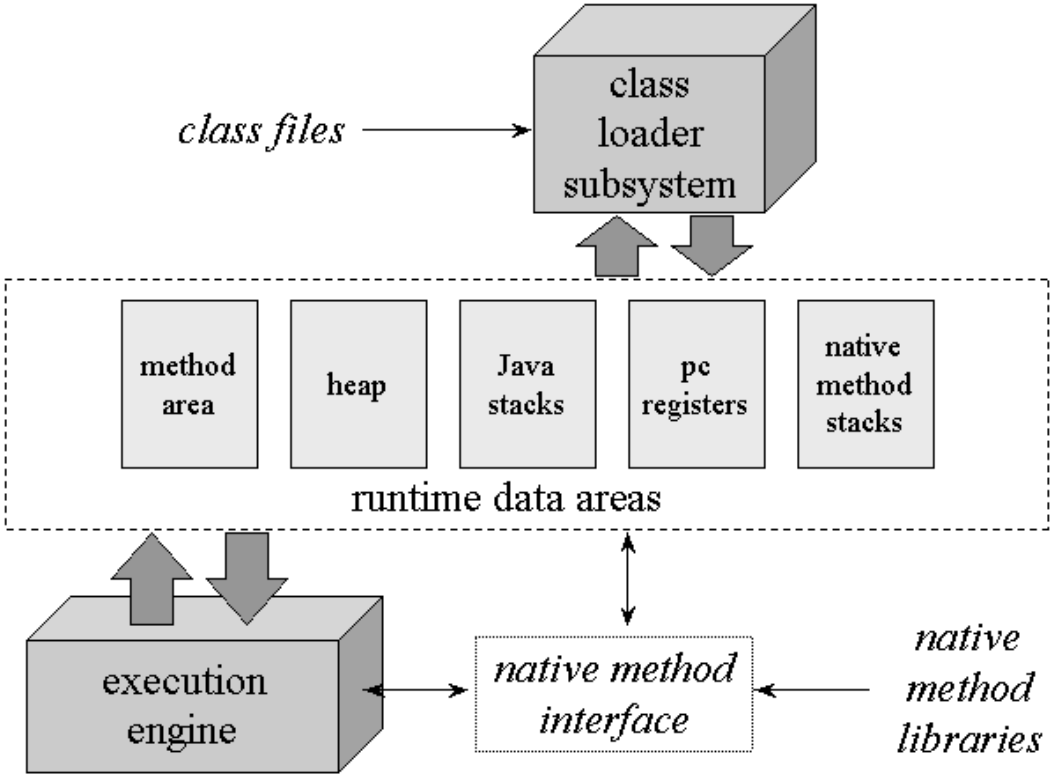
- Jej abstrakcyjnej specyfikacji – książki
- Konkretnej implementacji – oprogramowanie czasem sprzęt i oprogramowanie
- Instancji wykonywalnej – obsługującej program, istniejącą tak jak on

Są dwa rodzaje wątków (deamon i non-deamon).

Początek main() z jakiejś klasy.

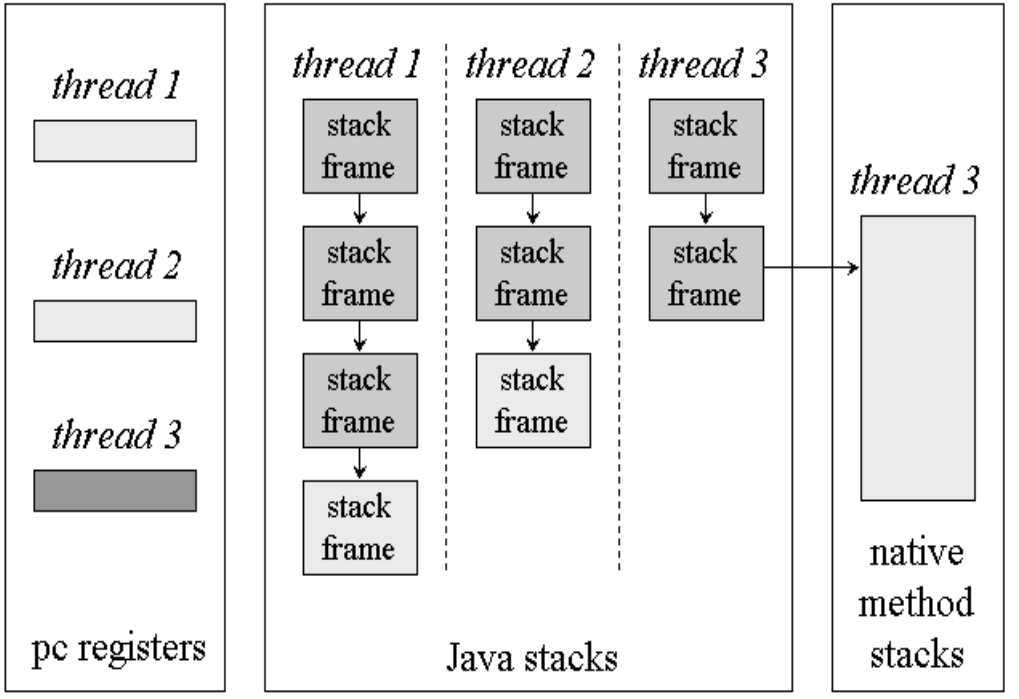
Koniec np. exit() z Runtime albo System

Budowa JVM


class files
class loader subsystem
method area
heap
Java stacks
pc registers
native method stacks
runtime data areas
execution engine
native method interface
native method libraries

Zarządzanie pamięcią

- Rodzaje obszarów pamięci
- Stos jest złożony z ramek
- Ramka zawiera stan wywołania jednej metody
- Nie używa się rejestrów



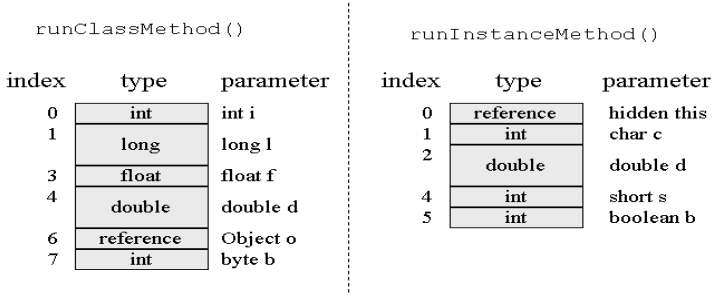
Obszary pamięci

Są różne podejścia do zarządzania pamięcią.

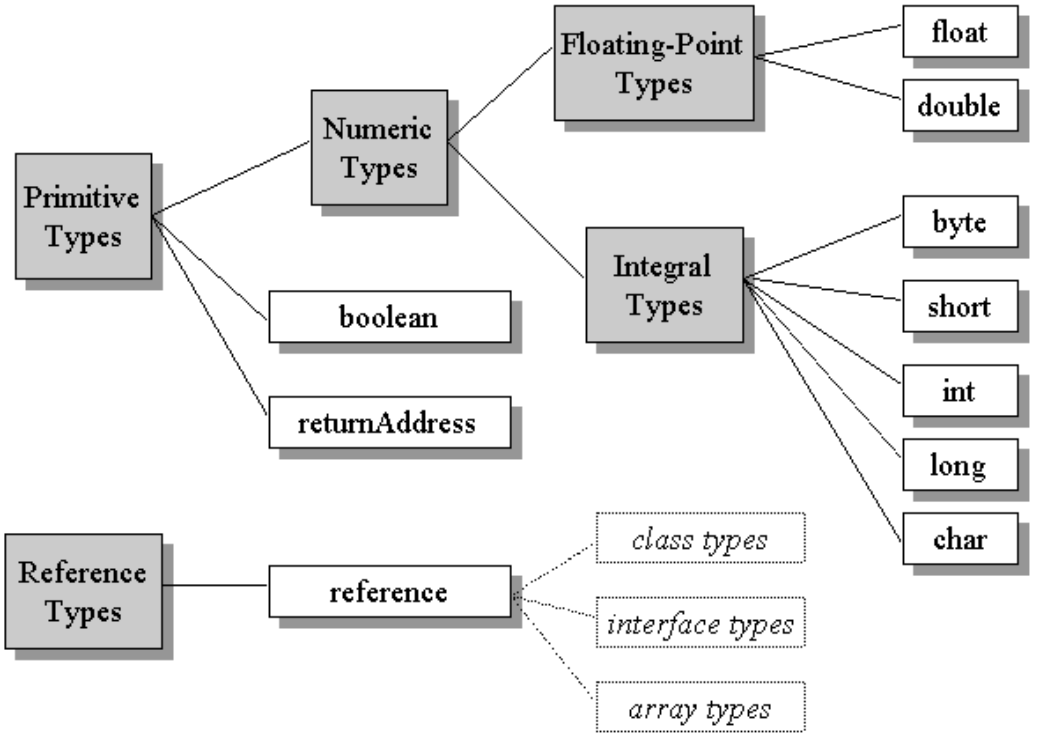
Obszary:

- Method area – uwaga na pojedyncze ładowanie klas
- Heap – różny dla aplikacji, dowolność zwalniania
- Java stack – różny dla wątków, przechowuje ramki
- Pc – trzyma „native pointer” i returnAddress
- Native method stacks – podobnie, ale dla metod „natywnych”

```
// On CD-ROM in file jvm/ex3/Example3a.java
class Example3a {
    public static int runClassMethod(int i, long l,
float f,
        double d, Object o, byte b) {
        return 0;    }
    public int runInstanceMethod(char c, double
d, short s,
        boolean b) {
        return 0;    }            }
```



Typy


Primitive Types
Numeric Types
Floating-Point Types
float
double
Integral Types
byte
short
int
long
char
boolean
returnAddress
Reference Types
reference
class types
interface types
array types

Podsystem ładowania klas

Są dwa rodzaje „ładowaczy” klas:

a) „bootstrap” – do stanowiących część JVM lub API

b) definiowane przez użytkownika – trzeba weryfikować ich poprawność. Wykonuje się kroki:

1) ładowanie

2) linkowanie (weryfikacja, przygotowanie)

3) inicjalizacja

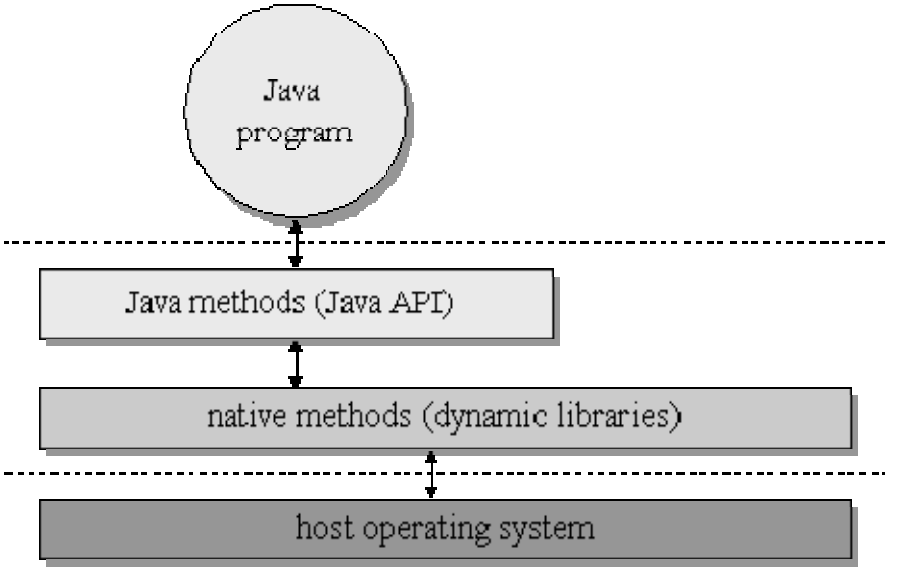
Silniki wykonawcze

Są trzy podejścia:

- Interpreter – instrukcje krok po kroku
- JIT – kompilacja do „native code” za pierwszym razem
- Adaptive optimizer – kompilacja tylko głównej części programu

„Native method interface”

- Można założyć dostępność pewnych klas w bibliotece
- Wpływ na bezpieczeństwo
- Wady przy przenośności wynikiem kompromisów



Ciekawostki

- 1995 pierwsza JVM – prosty interpreter
- Wady spowodowana swobodą implementacji
- Zwalnianie pamięci
- Ciekawe linki:

www.artima.com/insidejvm/ed2/
www.javaworld.com/javaworld/jw-02-1997/jw-02-hood.html
No i książka Eckela „Thinking in Java”