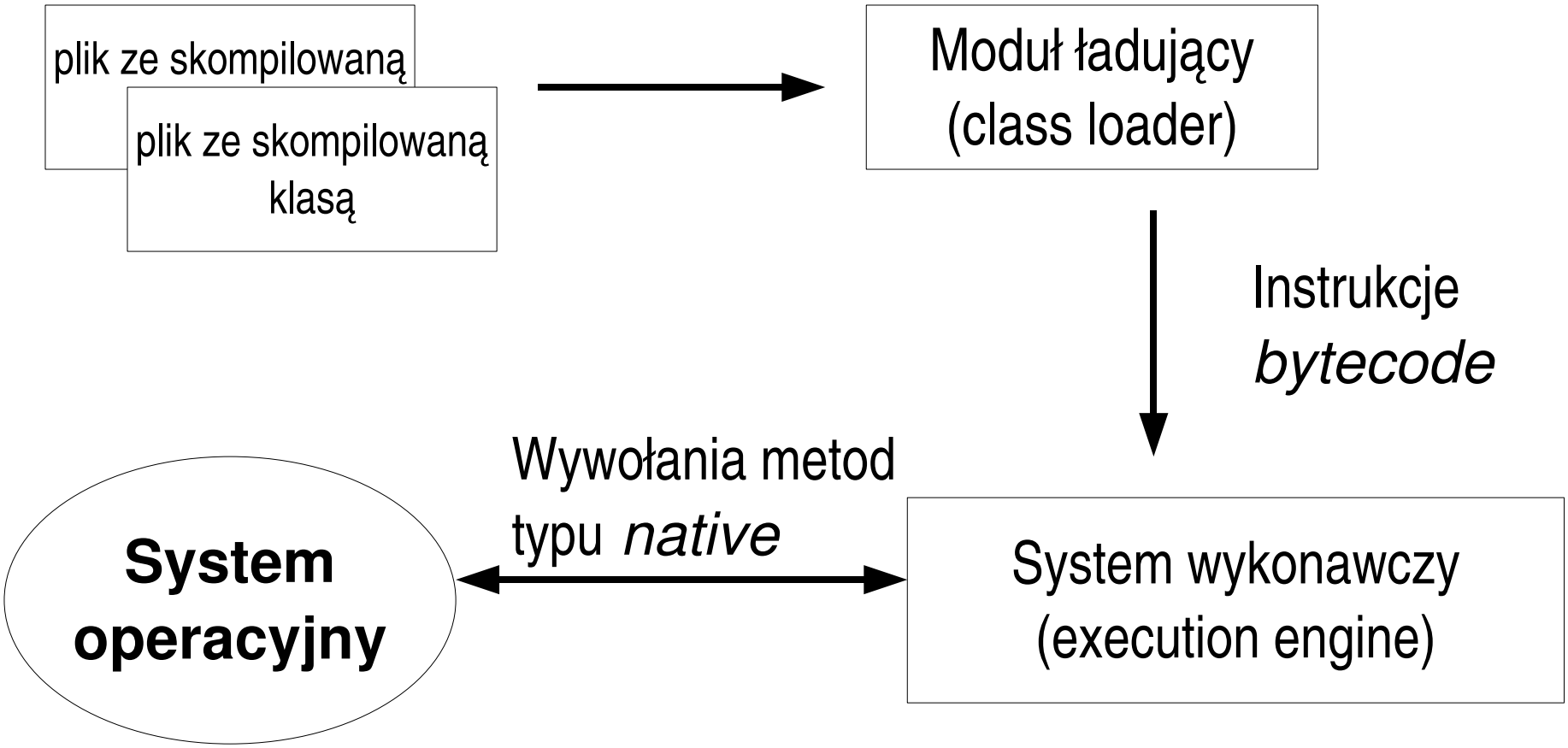
# Maszyna Wirtualna Javy

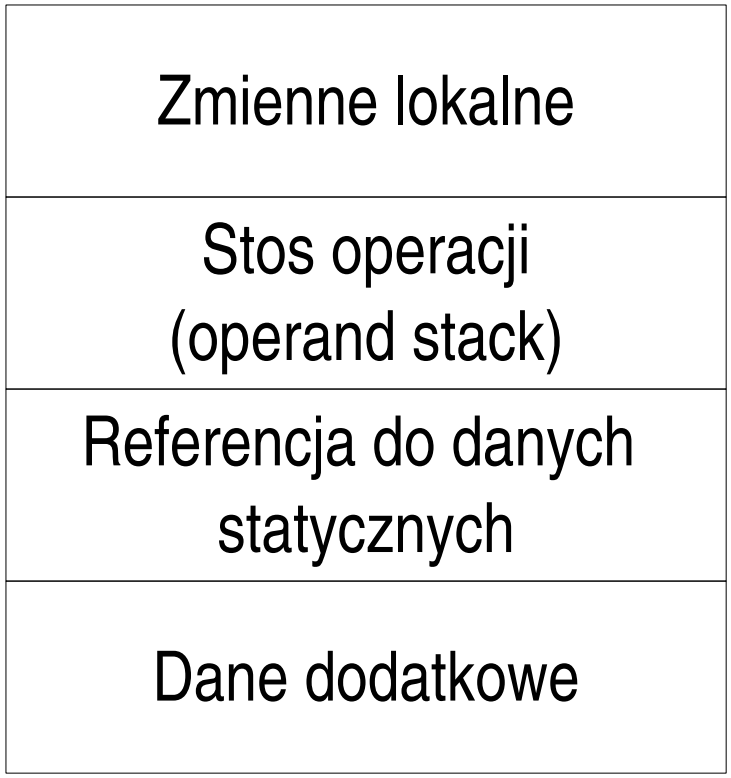
- Schemat ogólny



# System wykonawczy

- Typy:
  - Prosta maszyna wirtualna - interpreter wykonujący instrukcje *bytecode*
  - Kompilator "Just-In-Time" - przekształcenie przy pierwszym wywołaniu metody
  - Optymalizacja adaptacyjna (adaptive optimization)
- Obszar danych
  - "pc register" - licznik instrukcji (wątek)
  - JVM stack (wątek) - stos przechowujący ramki
  - heap (wspólna) - sterta do alokacji pamięci dynamicznej
  - method area (wspólny) - obszar kodu
  - runtime constant pool (klasa/interfejs) - obszar danych statycznych
  - native method stacks - stosy dla wywołań metod typu "native"

# Ramki (frame)




[referencja na obiekt] + [parametry] + zmienne lokalne
Zmienne typu *long* i *double* zajmują dwa miejsca

wszystkie dane potrzebne do wykonania pojedynczej instrukcji są ładowane tutaj

referencja do **runtime constant pool** dla klasy, do której należy metoda (w obszarze metod)

nie specyfikowane - możliwość implementacji własnych rozszerzeń (np. debugging)

```
public class MojaKlasa {
  public static int MetodaKlasy(int i,long l,float f,double d) {
            Object o=null; byte b=0;}
  public void Metoda(char c,short s){
             boolean b=FALSE;}}
```

# Instrukcje maszyny wirtualnej

- Każda instrukcja ma informację o typie argumentów (*iload, fadd, lstore*)
- Instrukcje przekazujące informacje pomiędzy stosem a obszarem danych
  - Ładuj zmienną na stos
  - Zapisz wartość ze stosu
  - Ładuj stałą na stos
- Instrukcje arytmetyczne
- Operacje na obiektach
  - Nowy obiekt lub tablica
  - Dostęp do pól obiektu/klasy (*getfield,putfield,getstatic,putstatic*)
  - Bezpośrednie operacje na stosie (operand stack)
- Kontrola programu (skoki warunkowe, bezwarunkowe, instrukcje wyboru )
- Wywołania metod: *invokevirtual, invokeinterface, invokespecial, invokestatic*
- Powrót z metody: *freturn, ireturn ...*

# Schemat działania maszyny dla pojedynczej instancji

- Klasa:
  - Ładowanie: znajdź binarną reprezentację wskazanej klasy i załaduj ją do pamięci
  - Konsolidacja (linking):
    - Weryfikacja - sprawdź poprawność załadowanej klasy
    - Przygotowanie - alokacja pamięci dla zmiennych klasy i wpisanie domyślnych wartości
    - Uzupełnienie referencji (resolution) - zamiana symbolicznych odnośników na rzeczywiste referencje
  - Inicjalizacja: Uruchomienie kodu Javy, który ustala początkowe wartości zmiennych klasy **<clinit>**
- Obiekt:
  - Alokacja pamięci i inicjalizacja danych
  - Wykonanie konstruktora (ewentualnie konstruktorów klas nadrzędnych) **<init>**

# Różne

- Ogólna struktura pliku z kodem pośrednim (class file)

| | |
|---|---|
| u4 magic;<br>u2 minor_version;<br>u2 major_version;<br>u2 constant_pool_count;<br>cp_info constant_pool[constant_pool_count-1];<br>u2 access_flags;<br>u2 this_class (wskaźnik na strukturę w cp);<br>u2 super_class (wskaźnik na nazwę w cp); | u2 interfaces_count;<br>u2 interfaces[interfaces_count];<br>u2 fields_count;<br>field_info fields[fields_count];<br>u2 methods_count;<br>method_info methods[methods_count];<br>u2 attributes_count;<br>attribute_info attributes[attributes_count]; |

- Tablice
- Dekompilacja kodu Javy i mechanizmy ukrywania (obfuscation)
  - przekształcanie identyfikatorów w kodzie (źródłowy lub pośredni)
  - uszkodzony kod pośredni
  - szyfrowanie kodu i użycie własnego programu ładującego