Programowanie Użytkowe

Dr. inż. Marcin Blachnik

Literatura

- Internet
- „Java Podstawy” – Horstmann & Cornell, Helion,
- „Java. Techniki Zaawansowane”, G.Cornell
- „Thinking in Java.”, Bruce Eckel
- „Java. Programowanie sieciowe”, Harold Elliotte R.

Co to Java

Java to:

- Język programowania
- Platforma programistyczna
- Platforma uruchomieniowa
- Zbiór różnych technologii

Popularność

- Dane wg. Tiobe Index:

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Apr 2016	Apr 2015	Change	Programming Language	Ratings	Change
1	1		Java	20.846%	+4.80%
2	2		C	13.905%	-1.84%
3	3		C++	5.918%	-1.04%
4	5	^	C#	3.796%	-1.15%
5	8	^	Python	3.330%	+0.64%
6	7	^	PHP	2.994%	-0.02%
7	6	v	JavaScript	2.566%	-0.73%
8	12	^^	Perl	2.524%	+1.18%

Cechy

Motto: Write once run everywhere

Języka:

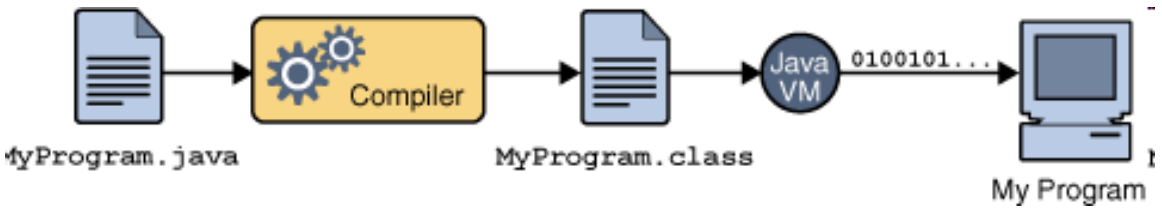
- Prosta
- Zorientowana obiektowo
- Bogate API
- Najbogatszym wsparcie różnych technologii

Platforma:

- Wydajna
- Wielowątkowa
- Bezpieczna
- Przenośna
- Rozproszona

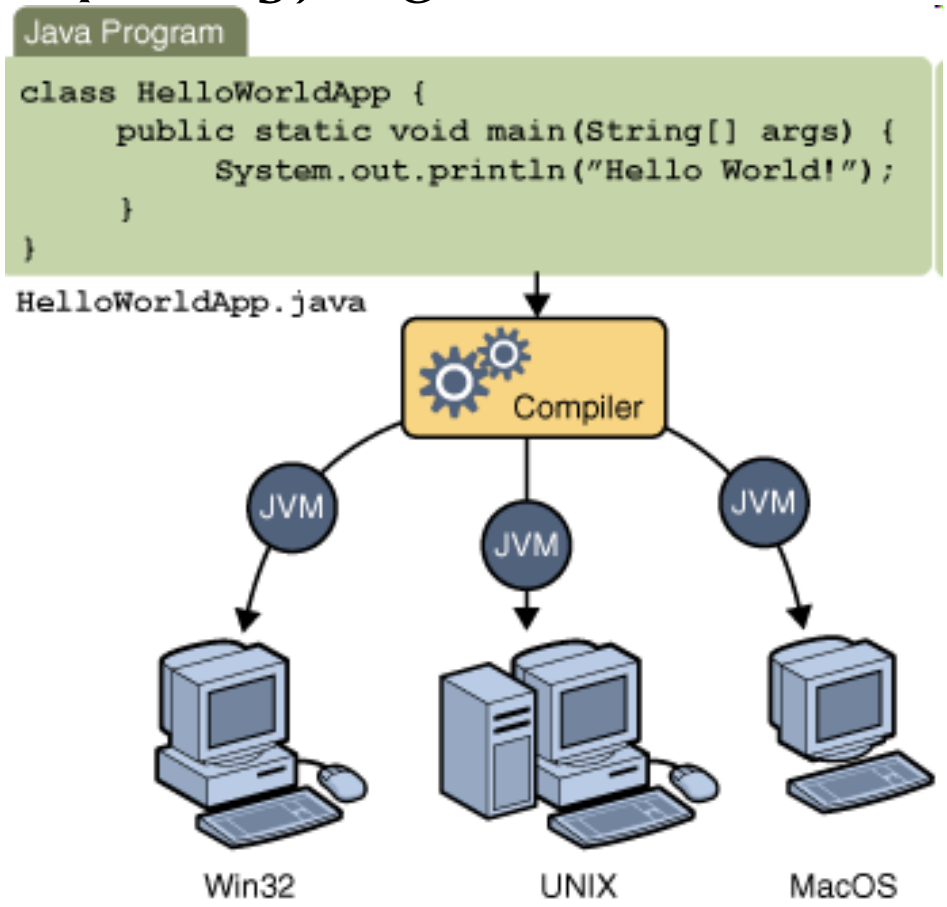
Filozofia Javy

- Kod źródłowy = pliki tekstowe z rozszerzeniem *.java
- Kompilacja za pośrednictwem kompilatora javac do plików *.class
- Pliki *.class zawierają „bytecodes” a nie natywny kod danego procesora!!!
- Aplikacja *.class uruchamiana za pomocą Java Virtual Machine = translacja kodu wewnętrznego *.class na kod maszynowy danego procesora



Java = niezależność

- Pliki skompilowane *.class są niezależne od systemu operacyjnego

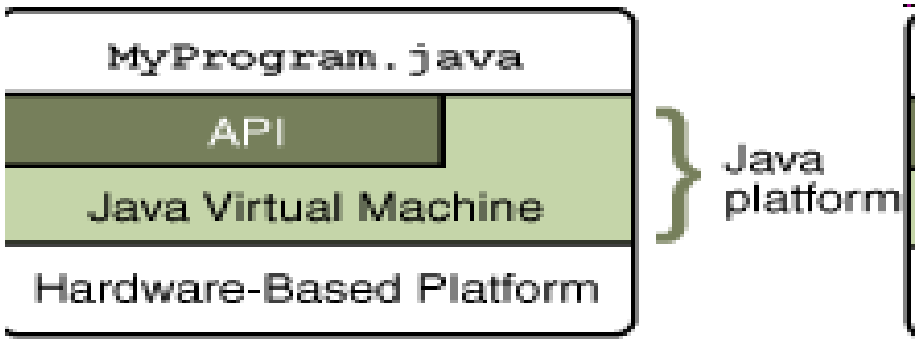


Platforma Javy

- Platforma – to sprzętowe lub softwareowe środowisko w którym uruchamiany jest program.
- Najpopularniejsze platformy: Microsoft Windows, Linux, Solaris OS, Mac OS (większość platform to kombinacja systemu operacyjnego i związanego z nim sprzętu)
- Platforma Javy to czysto softwareowa platforma działająca na bazie platformy sprzętowej i zewnętrznego systemu operacyjnego (choć nie koniecznie)
- Platforma Javy składa się z:
  - The Java Virtual Machine
  - The Java Application Programming Interface (API)

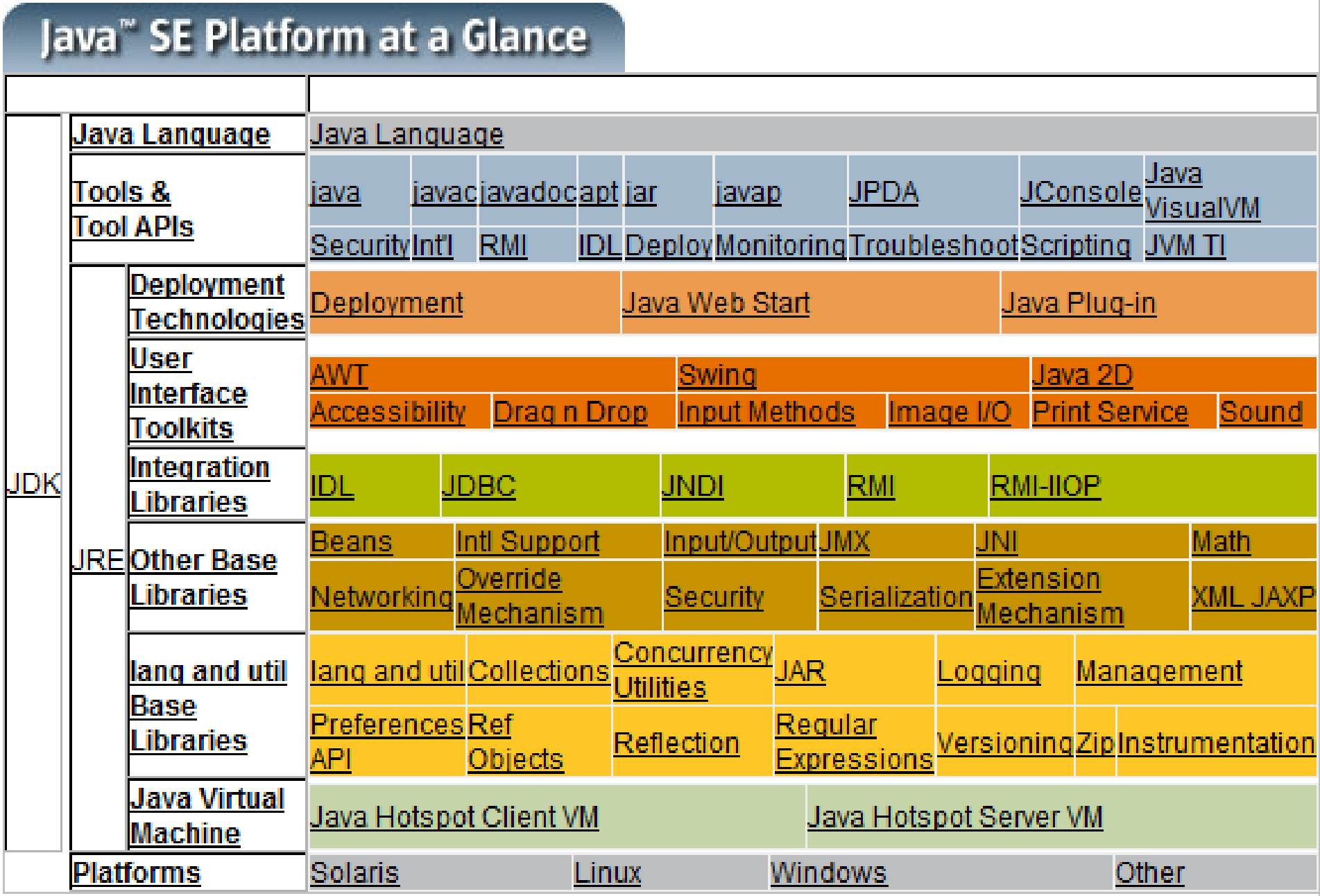
Platforma Javy

- API – jest rozbudowaną kolekcją gotowego oprogramowania (bibliotek) udostępniającą wiele użytecznych funkcji.
- API jest zbiorem pogrupowanych klas i interfejsów; owe biblioteki nazywane są pakietami



Oferta technologii Java

- **Narzędzia programistyczne**
  - Eclipse, NetBeans, IntelliJ, JDeveloper
- **Application Programming Interface (API)**: zbiór bibliotek
- **Technologie wdrożeniowe**: JDK udostępnia technologie typu Java Web Start, Java Plug-In w celu dostarczenia aplikacji do użytkownika końcowego
- **Narzędzia interfejsu użytkownika**: AWT + Swing oraz Java 2D jako narzędzia do tworzenia GUI + SWT (The Standard Widget Toolkit) + Java FX
- **Biblioteki integrujące**: Java IDL API, JDBC™ API, Java Naming and Directory Interface™ ("J.N.D.I.") API, Java RMI, Java Remote Method Invocation over Internet Inter-ORB Protocol Technology (Java RMI-IIOP Technology) udostępniają technologii dostępu do danych i zdalnego dostępu do obiektów., JNI – Java Native Interface etc.


Java™ SE Platform at a Glance
JDK
JRE
Java Language
Java Language
Tools & Tool APIs
java
javac
javadoc
apt
jar
javap
JPDA
JConsole
Java VisualVM
Security
Int'l
RMI
IDL
Deploy
Monitoring
Troubleshoot
Scripting
JVM TI
Deployment Technologies
Deployment
Java Web Start
Java Plug-in
User Interface Toolkits
AWT
Swing
Java 2D
Accessibility
Drag n Drop
Input Methods
Image I/O
Print Service
Sound
Integration Libraries
IDL
JDBC
JNDI
RMI
RMI-IIOP
Other Base Libraries
Beans
Intl Support
Input/Output
JMX
JNI
Math
Networking
Override Mechanism
Security
Serialization
Extension Mechanism
XML JAXP
lang and util Base Libraries
lang and util
Collections
Concurrency Utilities
JAR
Logging
Management
Preferences API
Ref Objects
Reflection
Regular Expressions
Versioning
Zip
Instrumentation
Java Virtual Machine
Java Hotspot Client VM
Java Hotspot Server VM
Platforms
Solaris
Linux
Windows
Other

JVM

- Wirtualna maszyna Javy
- Odpowiada za konwersję *bytocodu* do kodu wykonywalnego
- Konwersja realizowana jest przez JIT (Just in time)
- Konwersja wieloetapowa (zwykle co najmniej 3 etapy)
  - Etap 1 Interpreter + dodanie elementów wskaźników optymalizacyjnych
  - Etap 2 Analiza po kompilacji do kodu natywnego
  - Etap 3 Rekompilacja z optymalizacją w tym:
    - Inlining kodu, optymalizacja warunków, usuwanie martwego kodu, upraszczanie kodu, etc.
- Uwaga na wydajność – patrz:

http://www.mblachnik.pl/doku.php?id=notatki:informatyka:java

Garbage Collector

- GC - odśmiecacz zwalniający pamięć zajętą przez nieużywane już obiekty.
- Dzięki GC nowym obiektom wystarczy jedynie zaalokować pamięć i nie trzeba myśleć o zwalnianiu jej, ten proces dzieje się automatycznie przez wątek w którym działa GC.
- Wszystkie obiekty przechowywane są na wspólnej stercie zarządzanej, gdy są już nie potrzebne to są usuwane.

Jak działa GC

Podstawowe metody:

- Zliczanie referencji

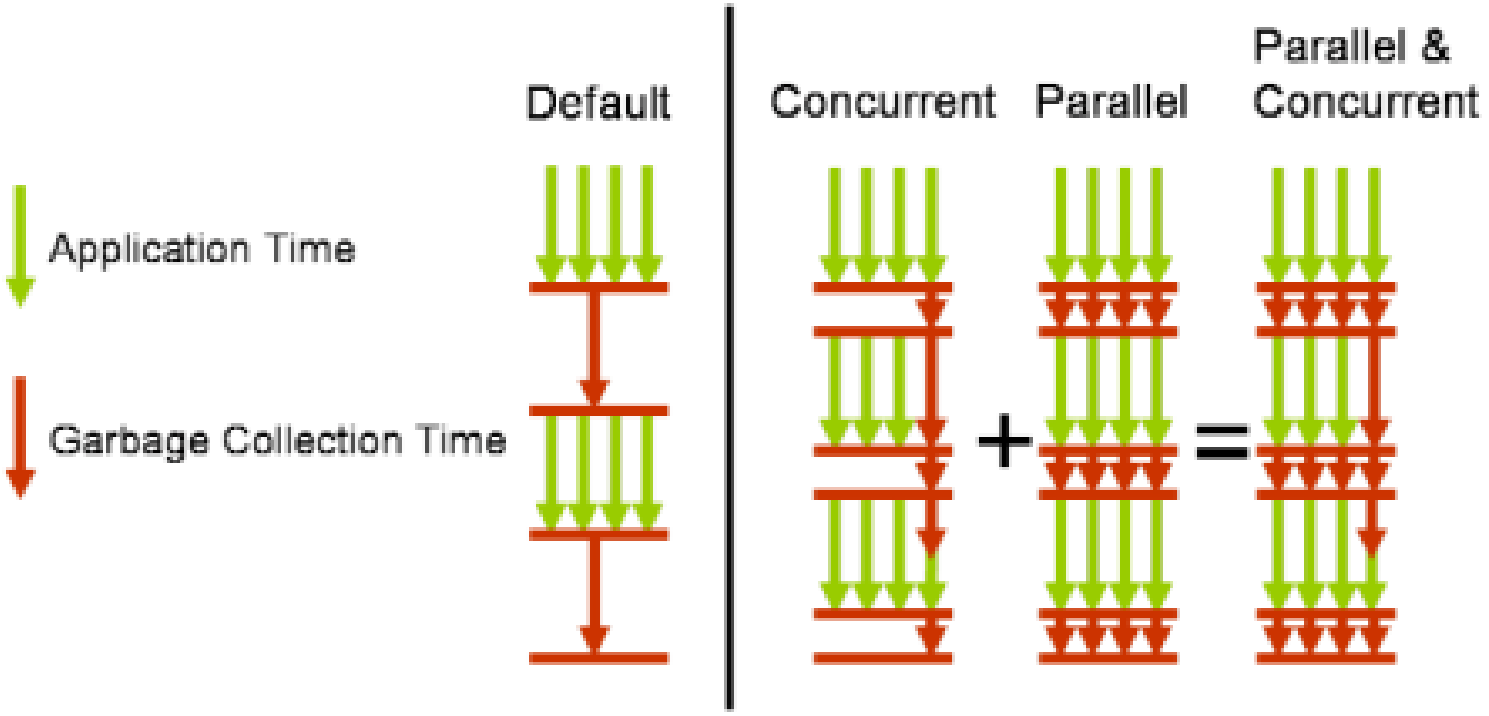
  każde stworzenie nowej referencji powoduje zwiększenie licznika referencji do danego obiektu, jeśli liczba referencji = 0 wówczas obiekt jako nie używany wyrzucany jest z pamięci. Wadą tej metody jest brak możliwości usunięcia obiektu w sytuacji w której obiekt A posiada referencję do obiektu B oraz obiekt B posiada referencję do obiektu A, wówczas w takiej sytuacji para obiektów A i B nigdy nie zostałaby usunięta.
- zaznacz i usuń (mark and sweep)

  każdy obiekt ma specjalny bit znacznika, który w momencie rozpoczęcia odśmiecania GC zeruje, następnie GC przechodzi stopniowo po strukturze obiektów od pnia w głąb i ustawia w/w bit we wszystkich obiektach które odwiedził. W końcowym etapie wszystkie obiekty z wyzerowanym bitem są usuwane jako nie podłączone.

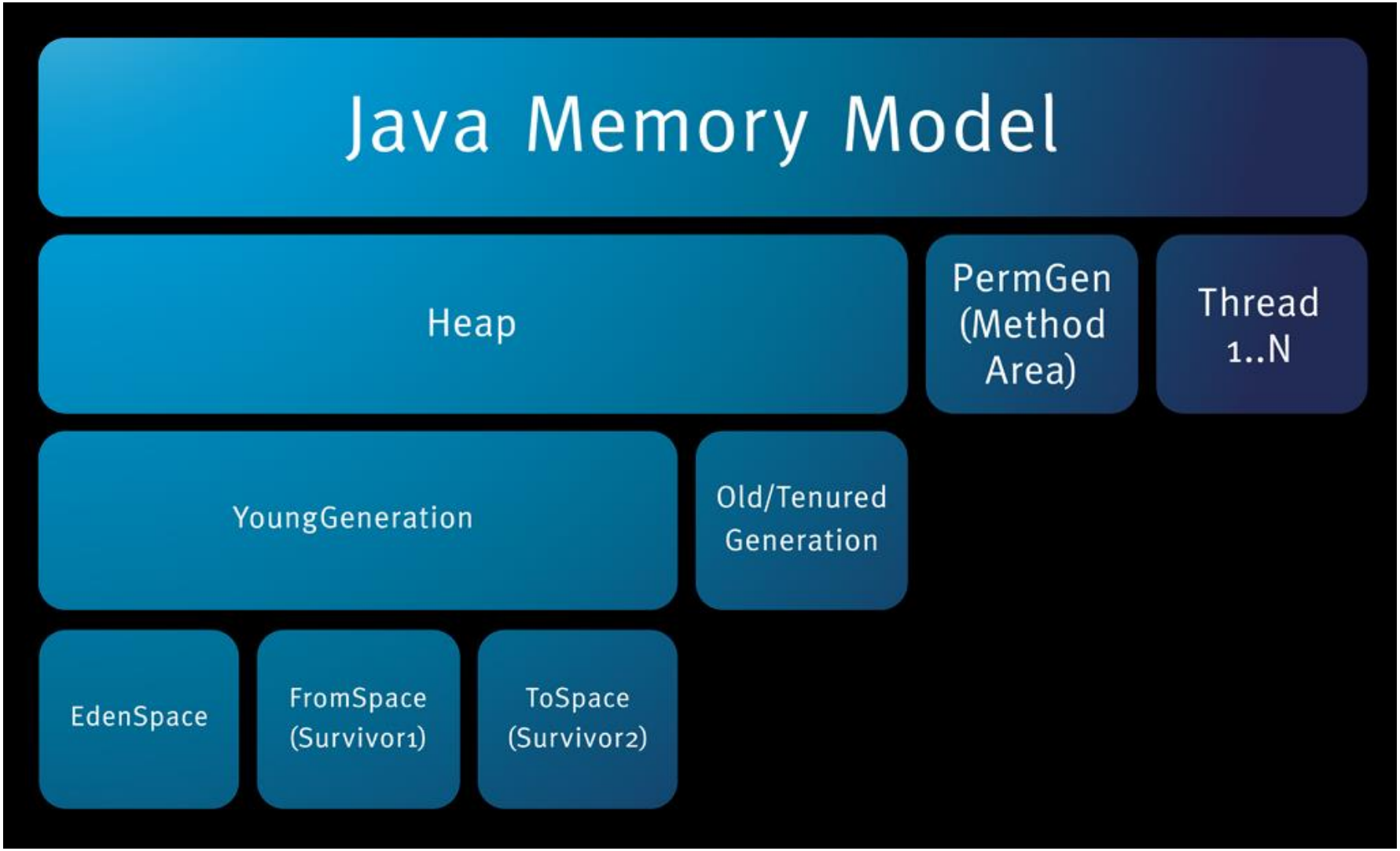
Co jeszcze nam daje GC

- Problem defragmentacji - w celu rozwiązania problemu defragmentacji pamięci po wyczyszczeniu pamięci z nieużywanych obiektów następuje proces kopiowania pozostałych w pamięci obiektów w nową lokalizację tak by zawsze dostępny był pełen i spójny obszar pamięci.

Jak działa GC



Model pamięci w Javie



UWAGA na GC

- GC nie opiekuje się zasobami (otwieranie i zamykanie dojść do plików, zamykanie strumieni, zamykanie sesji bazy danych itp..)
- GC nie zwalnia nas z zasad dobrego programowania - Możliwe wycieki pamięci!!!

Uruchamianie aplikacji

Uruchamianie aplikacji

- Java entrypoint – punkt wejścia do programu:
  - Dowolna klasa zawierająca metodę

public static void main(String[] args){}

- Java.exe / javaw.exe – narzędzia do uruchamiania programów
- Uruchomienie programu poprzez podanie nazwy klasy (klasa musi zawierać metodą *main*)

```
public class MojProgram {
   public static void main(String[] args) {
      System.out.println("Hello world");
   }
}
```

c:\Java\java.exe MojProgram

Uwaga jak uruchamiamy program to musimy być we właściwym katalogu, lub też podać właściwą ścieżkę w postaci „classpath”

Używanie plików JAR

- Jeśli archiwum jar to

  Java.exe –jar Plik.jar
- Co jeśli mamy kilka klas uruchomieniowych (kilka klas z metodami

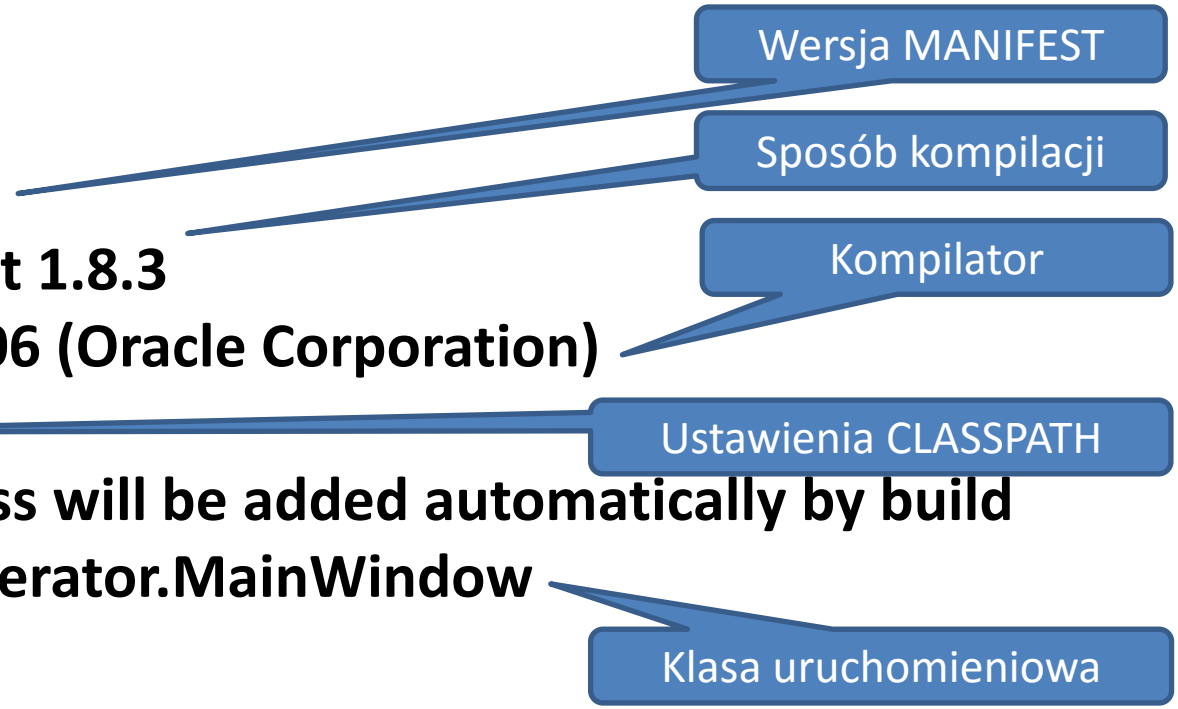
  public static void main(String[] args)
- Sami ręcznie podajemy klasę uruchamieniową np.

  java.exe –cp Plik.jar MojProgram
- Więc skąd java wie którą klasę uruchomieniową „odpalić”?
- W plikach „jar” katalog konfiguracyjny „META-INF”
  - Zawiera „MANIFEST.MF” – ustawienia konfiguracyjne podczas uruchamiania

Używanie plików JAR

- Przykład

Manifest-Version: 1.0
Ant-Version: Apache Ant 1.8.3
Created-By: 1.7.0_05-b06 (Oracle Corporation)
Class-Path:
X-COMMENT: Main-Class will be added automatically by build
Main-Class: randomgenerator.MainWindow



CLASSPATH

- CLASSPATH umożliwia definiowanie lokalizacji zewnętrznych bibliotek używanych w programie.
- Błędna konfiguracja CLASSPATH uniemożliwia poprawne działanie „naszej aplikacji”
- Możliwe sposoby definiowania:
  - Ustawienie zmiennej środowiskowej:

    SET CLASSPATH=
  - Ustawienia podczas uruchamiania

    java –cp biblioteki lub **java –classpath bibl**

bibl = „ścieżka1/biblioteka1.jar;ściezka2/biblioteka2.jar”

Parametry uruchomieniowe

- -Xms%Rozmiar_pamięci%m – początkowy rozmiar pamięci

  Np.. Java –Xms300m MojProgram

- -Xmx%Rozmiar_pamięci%m – maksymalny rozmiar pamięci
- -Dparametr=%wartość% - określenie właściwości środowiska uruchomieniowego

np. Java –Duser.home=„c:\Java” MojProgram

Inne parametry

- **-help** pomoc
- **-d32** jeśli możliwe wykorzystanie javy w wersji 32 bitowej
- **-d64** Jeśli możliwe wykorzystanie javy w wersji 64 bitowej
- **-version** Wyświetla informacje o wersji javy (uwaga na mikrowersje)
- **-server**uruchamia javę w wersji „serwer" zoptymalizowaną do aplikacji serwerowych (wymaga Javt z JDK).
- **-client** Wartość domyślna, uruchamia javę w wersji „klient" zoptymalizowaną do aplikacji z GUI, dowolna wersja JRE
- **-verbosegc** Wyświetlenie informacji o uruchamianiu się GC
- **-prof:java.prof** Zapis danych do analizy wydajności do pliku .\java.prof.
- **-XX:kkkk** Różne przełączniki np. **-XX:+UseG1GC** – uruchomienie GC typu G1
- **-XX:MaxPermSize=64m** Rozmiar pamięci: *permanent generation* memory pool liczony w MB dla długo żyjących obiektów
- **-XX:+UseCompressedOops** Wykorzystanie skompresowanych referencji dla Javy 64bitowej jeśli XMX < 32GB – mniejsze zużycie pamięci, szybsze działanie aplikacji .
- **-XX:+UseParNewGC** Wykorzystanie współbieżnego GC

UWAGA
Java class format

- Podczas kompilacji każda klasa zawiera informację o użytym kompilatorze (wersji Javy)
- Jeżeli główny program uruchomiono w wersji javy M, a biblioteka jest w wersji N, tak że M<N to pojawi się błąd (ważne dla autorów aplikacji serwerowych, bo trzeba dopasować się do ustawień serwera)

major	minor	Java platform version
45	3	1.0
45	3	1.1
46	0	1.2
47	0	1.3
48	0	1.4
49	0	1.5
50	0	1.6

- Jak sprawdzić wersję klasy

  javap – dekompilator

  Np.. Javap –verbose MojProgram