



Wprowadzenie do technologii JVM

Mateusz Duraj

Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy



1. JVM
2. Byte code
3. ClassLoader
4. JIT
5. Java memory model
6. GC & Memory Leaks
7. JDK Tools



- JVM to akronim dla Java Virtual Machine, tj. Wirtualna Maszyna Javy.
- Aby aplikacja napisana w języku Java mogła być uruchomiona, pliki zawierające kod źródłowy muszą być skompilowane.
- Kompilacja polega na przetłumaczeniu programu napisanego w Javie na kod wykonywalny, tzw. Bytecode.
- Kompilację wykonujemy przy użyciu specjalnego narzędzia, tzw. kompilatora.



- JDK
- JRE
- JVM

JDK

`javac, jar, debugging tools,
javap`

JRE

`java, javaw, libraries,
rt.jar`

JVM

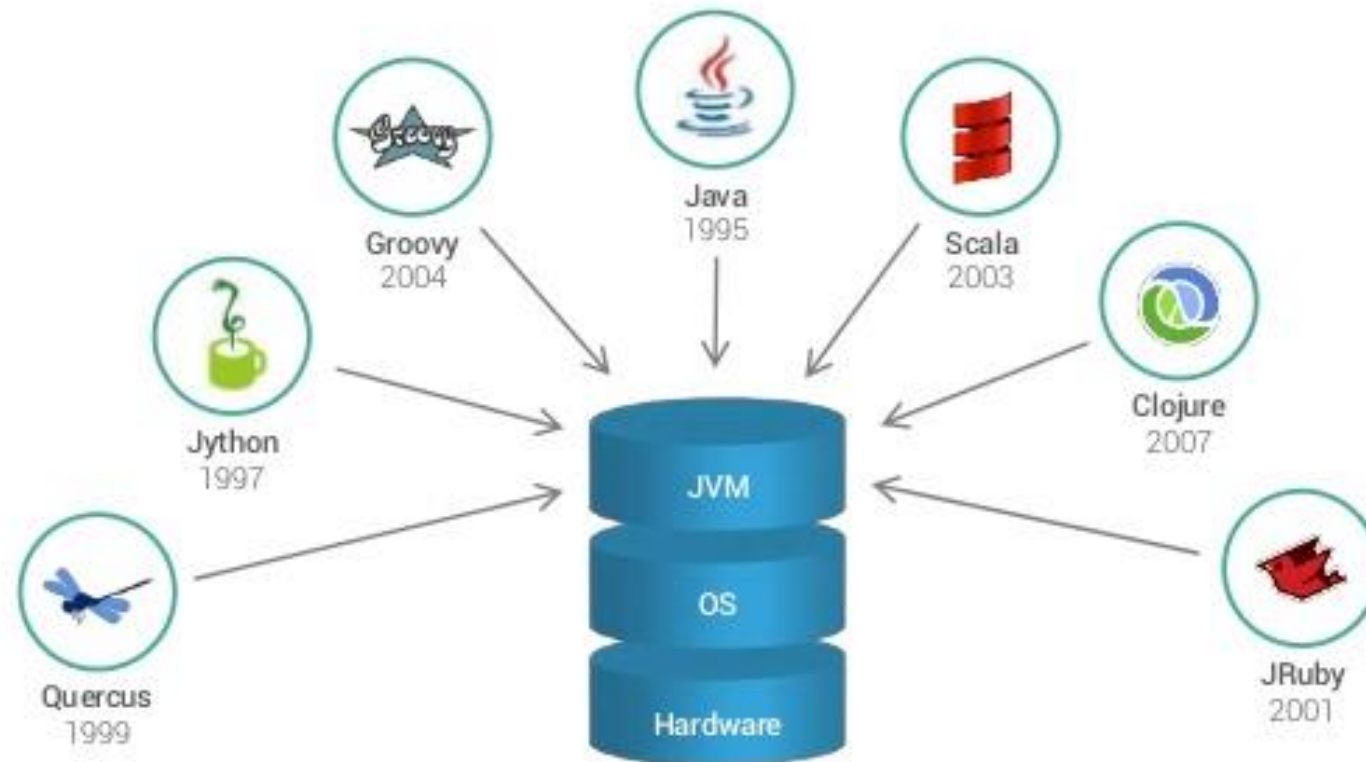
Just In Time
Compiler (JIT)

JVM – co to takiego ?



Scala Fundamentals

Java and the JVM



Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy



HotSpot – oficjalna dystrybucja, od 11 płatna dla użytku komercyjnego

OpenJDK – równoległe rozwijana przez Oracla

DoppioJVM - napisana w TypeScript, można odpalać w przeglądarce

leJOS – maszyn wirtualna dla Lego Mindstorms 😊



- Maszyna HotSpot bazuje w dużej mierze na OpenJDK
- Zawiera dodatkowo np. Java Flight Recorder (profiler dla javy) w wersji komercyjnej



source -> javac -> bytecode

bytecode -> classloader -> interpreter

interpreter -> JIT -> optimized native code

Zadanie 1



Pobierz projekt jvm2 z repo

Zweryfikuj nazwę implementacji JVM zainstalowanej na Twoim systemie.
W tym celu wykorzystaj properties:

- java.vm.name
- java.home
- java.version
- java.vendor
- java.specification.vendor

Wartość w/w kluczy wypisz na konsolę (otwórz klasę **JvmConfig**)



- Kod maszynowy
https://en.wikipedia.org/wiki/Java_bytecode_instruction_listings
- Instrukcje
 - ładujące i zapisujące (np. aload_0, istore)
 - arytmetyczne i logiczne (np. ladd, fcmpl)
 - zmieniające typ (np. i2b, d2i)
 - tworzące obiekt i manipulujące nim (np. new, putfield)
 - przenoszące sterowanie (np. ifeq, goto)
 - wywołujące metodę i wracające z niej (np. invokespecial, areturn)



Zadanie 2 Generowanie kod bajtowego

- W klasie JavapTrain:
- Z poziomu CMD(Windows) wejdź do katalogu projektu(package pl.sdacademy.jvm)
- Wykonaj polecenie `javac JavapTrain.java`
- Wykonaj polecenie `javap -c JavaTrain.class > JavaTrain.bc`
- Dla poleceń `javac` i `javap` sprawdź dostępne opcje z użycie parametru `-help`



- Zawiera meta informacje np. wersje javy, superclass, interfaces...
- Informacje przechowywane są w formacie HEX
- Szczegółowy opis formatu pliku.class:
<https://codecouple.pl/2018/06/15/co-kryje-plik-class/>



.class wersja HEX

```
00000000: cafe babe 0000 0033 001d 0a00 0600 0f09 .....3.....
00000010: 0010 0011 0800 120a 0013 0014 0700 1507 .....
00000020: 0016 0100 063c 696e 6974 3e01 0003 2829 .....<init>...()
00000030: 5601 0004 436f 6465 0100 0f4c 696e 654e V...Code...LineN
00000040: 756d 6265 7254 6162 6c65 0100 046d 6169 umberTable...mai
00000050: 6e01 0016 285b 4c6a 6176 612f 6c61 6e67 n...([Ljava/lang
00000060: 2f53 7472 696e 673b 2956 0100 0a53 6f75 /String;)V...Sou
00000070: 7263 6546 696c 6501 0009 4d61 696e 2e6a rceFile...Main.j
00000080: 6176 610c 0007 0008 0700 170c 0018 0019 ava.....
00000090: 0100 0c48 656c 6c6f 2057 6f72 6c64 2107 ...Hello World!.
000000a0: 001a 0c00 1b00 1c01 0004 4d61 696e 0100 .....Main..
000000b0: 106a 6176 612f 6c61 6e67 2f4f 626a 6563 .java/lang/Objec
000000c0: 7401 0010 6a61 7661 2f6c 616e 672f 5379 t...java/lang/Sy
000000d0: 7374 656d 0100 036f 7574 0100 154c 6a61 stem...out...Lja
000000e0: 7661 2f69 6f2f 5072 696e 7453 7472 6561 va/io/PrintStrea
000000f0: 6d3b 0100 136a 6176 612f 696f 2f50 7269 m;...java/io/Pri
00000100: 6e74 5374 7265 616d 0100 0770 7269 6e74 ntStream...print
00000110: 6c6e 0100 1528 4c6a 6176 612f 6c61 6e67 ln...([Ljava/lang
00000120: 2f53 7472 696e 673b 2956 0021 0005 0006 /String;)V.!....
00000130: 0000 0000 0002 0001 0007 0008 0001 0009 .....
00000140: 0000 001d 0001 0001 0000 0005 2ab7 0001 .....*...
00000150: b100 0000 0100 0a00 0000 0600 0100 0000 .....
00000160: 0100 0900 0b00 0c00 0100 0900 0000 2500 .....%.
00000170: 0200 0100 0000 09b2 0002 1203 b600 04b1 .....
00000180: 0000 0001 000a 0000 000a 0002 0000 0004 .....
00000190: 0008 0005 0001 000d 0000 0002 000e .....
```

Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy



Ładuje dynamicznie klasy

JVM mówi że „potrzebuje takiej klasy”

Czyli mamy pierwszy etapy source->javac->bytecode

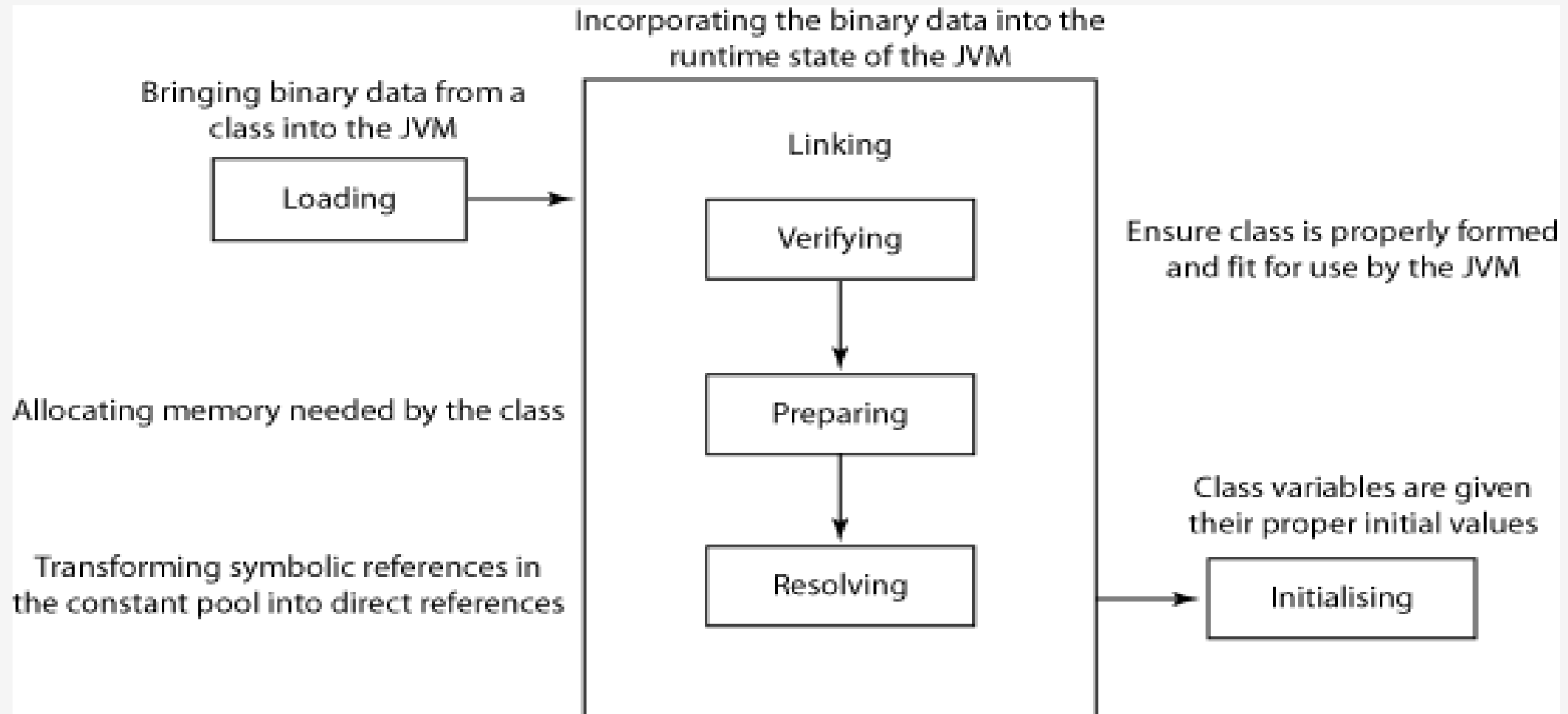
Kolejny : classloader -> ... ?



- Ładowanie – odczyt plików .class
- Linkowanie
 - Weryfikacja poprawności kodu bajtowego (czy np. nie rozszerzamy klasę final)
 - Alokowanie pamięci (czyli np. w klasie mamy tablice)
 - Wiązanie obiektów (np. mamy dziedziczenie)
- Inicjalizacja statycznych pól klasy



CLASSLOADER - FAZY





CLASSLOADER – faza Loading

- Ładowanie kodu bajtowego do pamięci
- Lokalizacja innych klasy (na podstawie classpaths)
- Bootstrap class loader dla ładowania klas z rt.jar
- Extension class loader dla ładowania klas z \$JAVA_HOME/lib/ext
- System class loader dla ładowania klas dla naszej aplikacji (classpath)
- Na tym etapie może pojawić się ClassNotFoundException, NoClassDefFoundError



- Weryfikacja poprawności kodu bajtowego (czy np. nie rozszerzamy klasę final)
- Alokowanie pamięci (czyli np. w klasie mamy tablice)
- Wiązanie obiektów (np. mamy dziedziczenie)
- Sprawdzanie poziomów dostępu



CLASSLOADER – faza Initialization

- Wykonanie metod inicjalizacyjnych na klasach i interfejsach
- Inicjowanie pól ich wartościami domyślnymi
- Inicjowanie super klas, jeżeli do tej pory nie zostały zainicjowane
- Obiekt klasy jest gotowy do użycia w programie

Zadanie 3 ClassLoader



- Z poziomu cmd otwórz katalog z klasą **ClassLoader**
- Wykonaj polecenie `java -verbose ClassLoader`

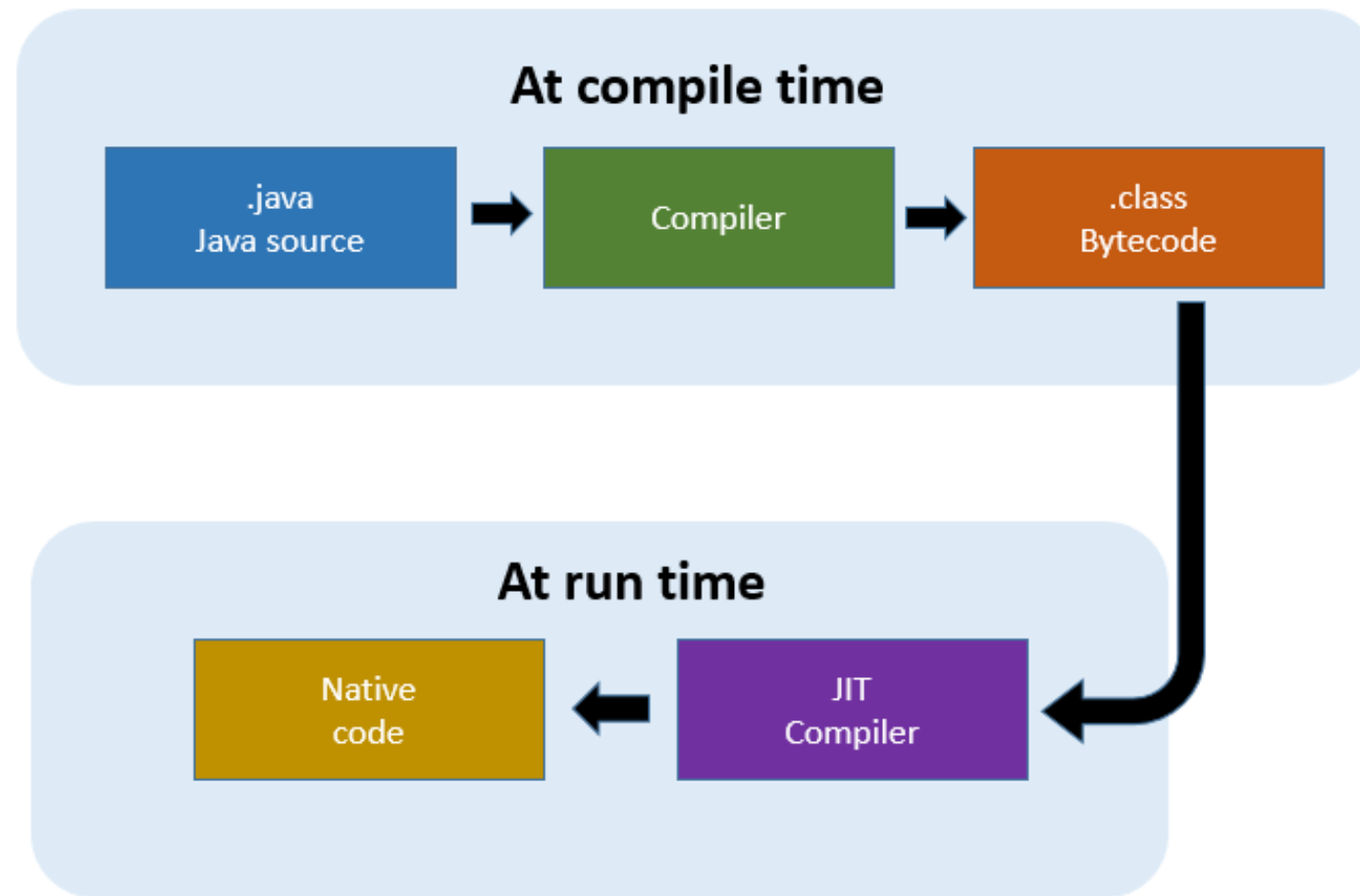


- Java jest językiem interpretowanym , a więc kompilowany do bytecode, a potem jest to interpretowane przez JVM
- Hot spot – interpreter javy cały czas analizuje kod i np. weryfikuje jak często wywoływana jest dana metoda



- Kod po fazie „interpretowania”, jest analizowany poprzez wbudowany mechanizm optymalizacji jit
- Średnio jit potrafi zoptymalizować kod do 20 razy szybciej(przy konwersji do kodu natywnego)
- JIT działa w wersjach client i serwer (determinuje to konfiguracja sprzętowa)

JIT (Just-In-Time)



Zródło: <https://aboullaite.me/understanding-jit-compiler-just-in-time-compiler/>

Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy



- Inlining (zagnieżdżanie metod)
- Loop unrolling
- Dead-code elimination (eliminacja martwego kodu)



JIT (Just-In-Time) Inlining

```
public String getValueFromSupplier(Supplier < String > supplier) {  
    return supplier.get();  
}
```

```
public String businessMethod(String param) {  
    Supplier < String > stringSupplier = () -> ("my" + param);  
    return getValueFromSupplier(stringSupplier);  
}
```

// zostanie zamieniona na

```
public String businessMethod(String param) {  
    Supplier < String > stringSupplier = () -> ("my" + param);  
    return stringSupplier.get();  
}
```

|



JIT (Just-In-Time) Loop unrolling

```
for (int i = 0; i < N; i++) {  
    S(i);  
}
```

// Zostanie zamienione na

```
for (int i = 0; i < N; i += 4) {  
    S(i);  
    S(i + 1);  
    S(i + 2);  
    S(i + 3);  
}
```



JIT (Just-In-Time) Dead code elimination

```
public String businessMethod(String param) {  
    Supplier < String > supplier = new StringSupplier("my" + param);  
    if (supplier == null) {  
        throw new IllegalArgumentException("Supplier cannot be null");  
    }  
    return supplier.get();  
}
```

// Zostanie zamienione na

```
public String businessMethod(String param) {  
    Supplier < String > supplier = new StringSupplier("my" + param);  
    return supplier.get();  
}
```



- `-XX:+UnlockDiagnosticVMOptions`
- `-XX:+LogCompilation - hotspot_pid.log` – drukuje informacje o poziomach client/serwer
- `-XX:+TraceClassLoading`



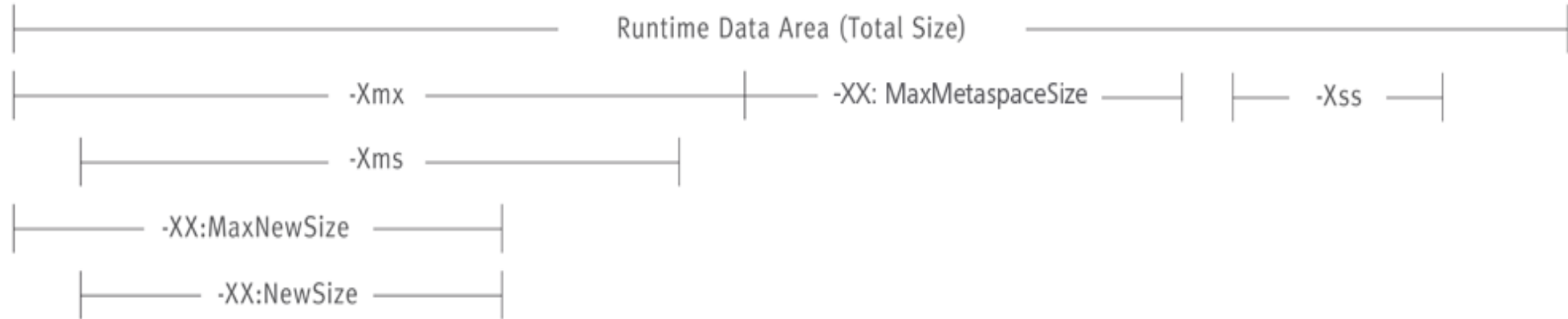
Java jest językiem kompilowany czy interpretowanym ?



- Dlaczego klasa String jest final ?
- Różnica pomiędzy `new String("Message ")` a `"Message"`



JAVA MEMORY MODEL



Heap Space						Metaspace		Native Area					
Young Generation				Old Generation		Permanent Generation		Code Cache					
Virtual	From Survivor 0	To Survivor 1	Eden	Tenured	Virtual	Runtime Constant Pool	Virtual	Thread 1..N			Compile	Native	Virtual
						Field & Method Data		PC	Stack	Native Stack			
						Code							

Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy



Jak działa pamięć w javie – stos

```
public class Main
{
    ➔ public static void main(String[] args)
    {
        int value = 7;
        value = calculate(value);
    }

    public static int calculate(int data)
    {
        int tempValue = data + 3;
        int newValue = tempValue * 2;
        return newValue;
    }
}
```





Jak działa pamięć w javie – stos

```
public class Main
{
    public static void main(String[] args)
    {
        int value = 7;
        value = calculate(value);
    }

    public static int calculate(int data)
    {
        int tempValue = data + 3;
        int newValue = tempValue * 2;
        return newValue;
    }
}
```

args = empty
array

Jak działa pamięć w javie – stos



```
public class Main
{
    public static void main(String[] args)
    {
        int value = 7;
        value = calculate(value);
    }

    public static int calculate(int data)
    {
        int tempValue = data + 3;
        int newValue = tempValue * 2;
        return newValue;
    }
}
```



value = 7

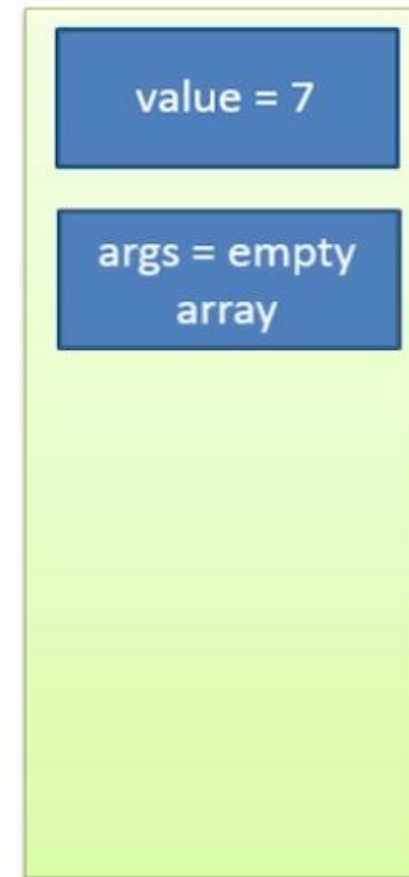
args = empty
array

Jak działa pamięć w javie – stos



```
public class Main
{
    public static void main(String[] args)
    {
        int value = 7;
        value = calculate(value);
    }

    public static int calculate(int data)
    {
        int tempValue = data + 3;
        int newValue = tempValue * 2;
        return newValue;
    }
}
```

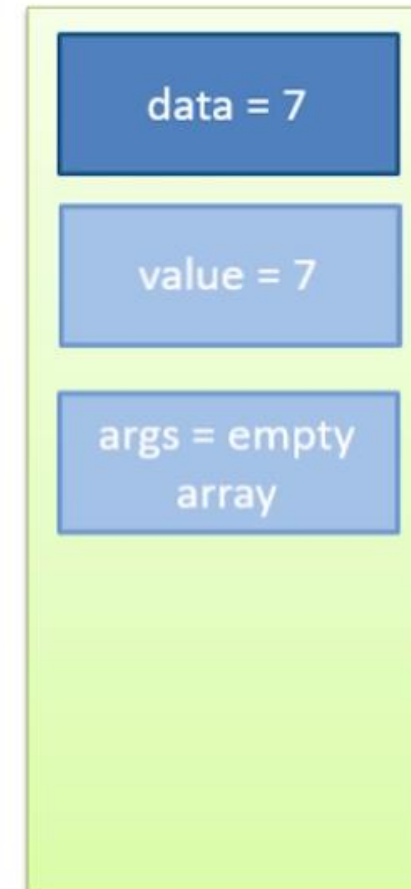





Jak działa pamięć w javie – stos

```
public class Main
{
    public static void main(String[] args)
    {
        int value = 7;
        value = calculate(value);
    }

    public static int calculate(int data)
    {
        int tempValue = data + 3;
        int newValue = tempValue * 2;
        return newValue;
    }
}
```

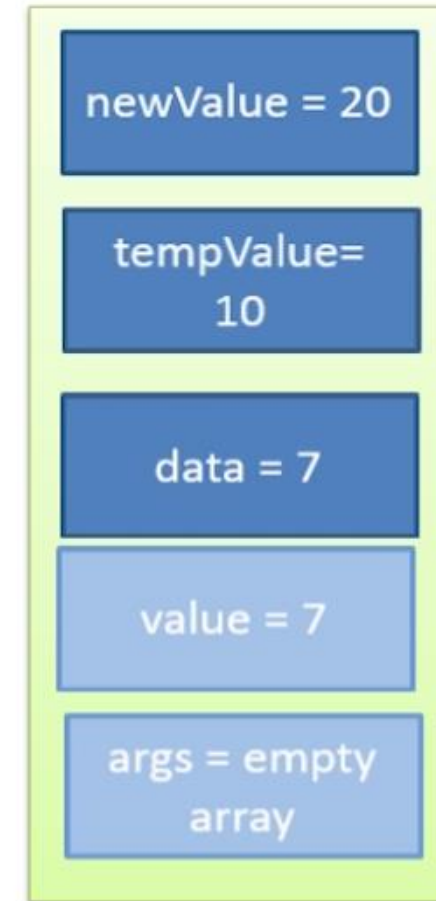






Jak działa pamięć w javie – stos

```
public class Main
{
    public static void main(String[] args)
    {
        int value = 7;
        value = calculate(value);
    }

    public static int calculate(int data)
    {
        int tempValue = data + 3;
        int newValue = tempValue * 2;
        return newValue;
    }
}
```






Jak działa pamięć w javie – stos

```
public class Main
{
    public static void main(String[] args)
    {
        int value = 7;
        value = calculate(value);
    }

    public static int calculate(int data)
    {
        int tempValue = data + 3;
        int newValue = tempValue * 2;
        return newValue;
    }
}
```



value = 20

args = empty
array

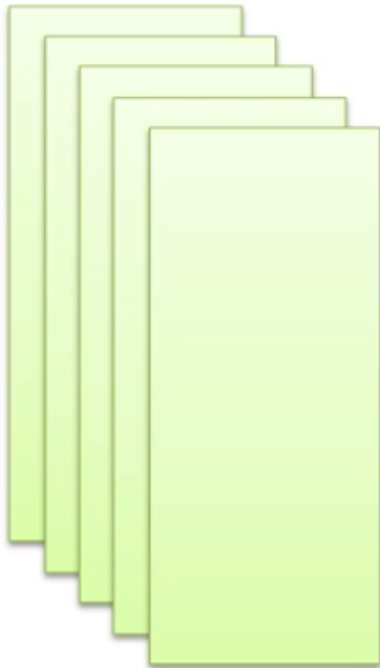


Maszyna wirtualna Javy startuje zapewniając sobie miejsce na stertę w pamięci RAM (ang. *Random Access Memory*). Mówimy, że alokuje tę pamięć. Na przykład jeśli Twój komputer ma 4GB pamięci RAM maszyna wirtualna podczas startu może przydzielić sobie 512MB. Obszar ten potencjalnie może się powiększyć w zależności od ustawień maszyny wirtualnej i dostępnej pamięci.

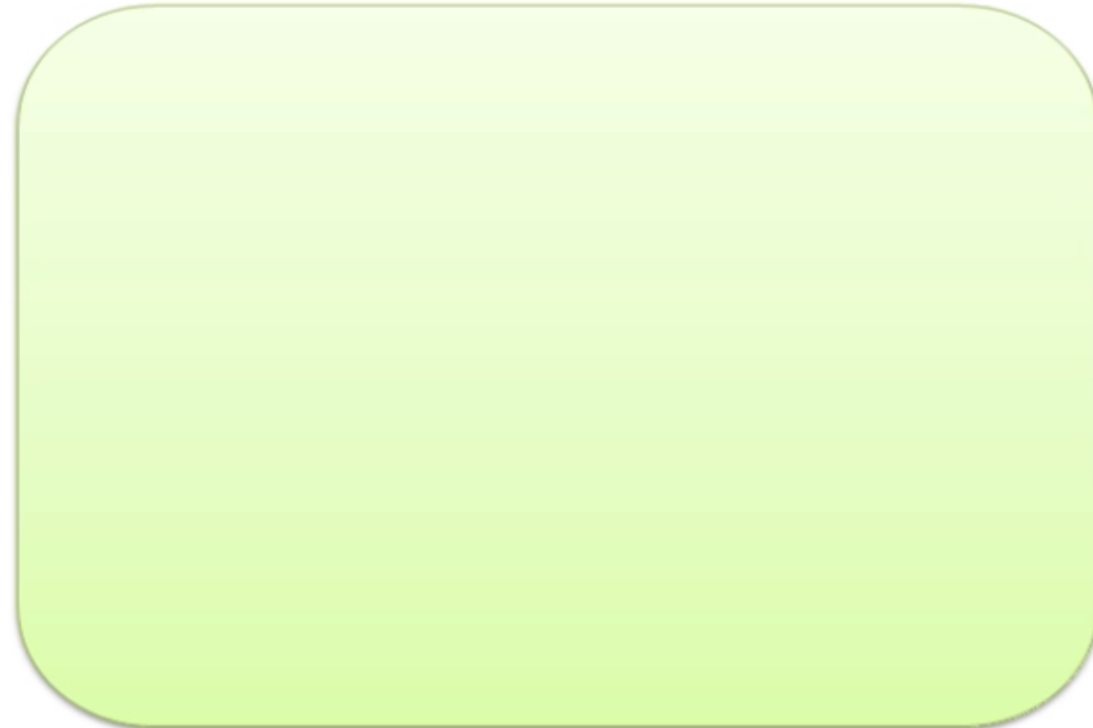
Jak działa pamięć w javie – sarta



The Stack



The Heap



Autor: Mateusz Duraj

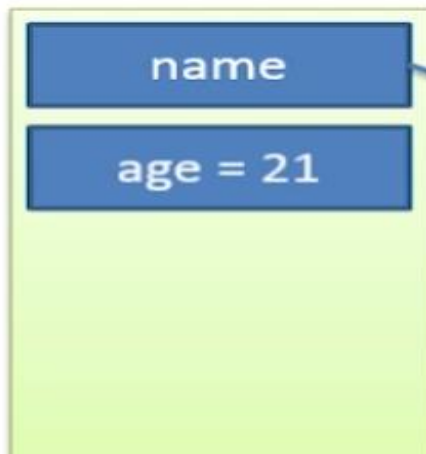
Prawa do korzystania z materiałów posiada Software Development Academy



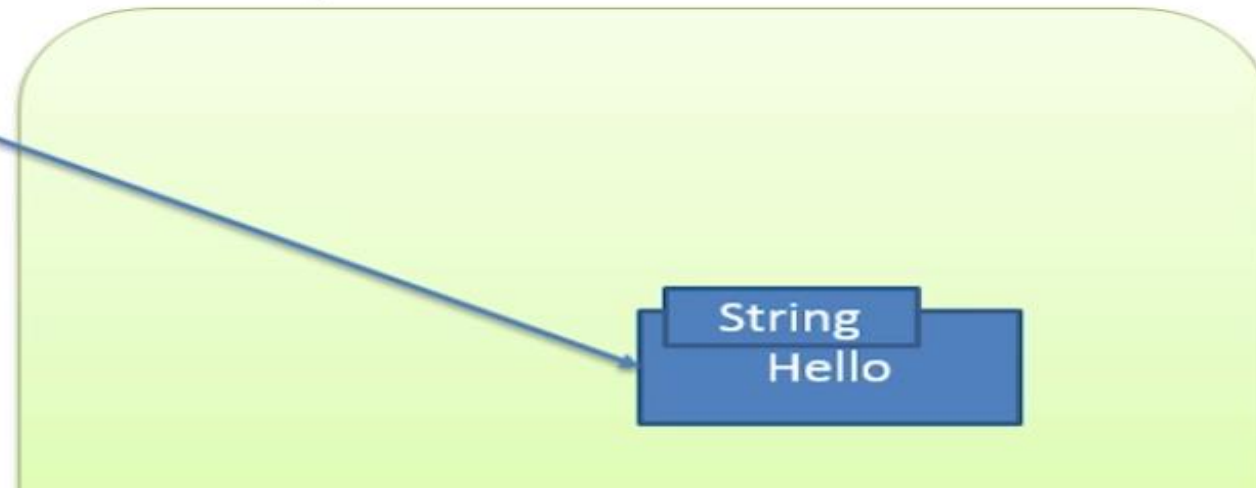
Jak działa pamięć w javie – stereta

```
int age = 21;  
String name = "Hello";
```

Stack



Heap

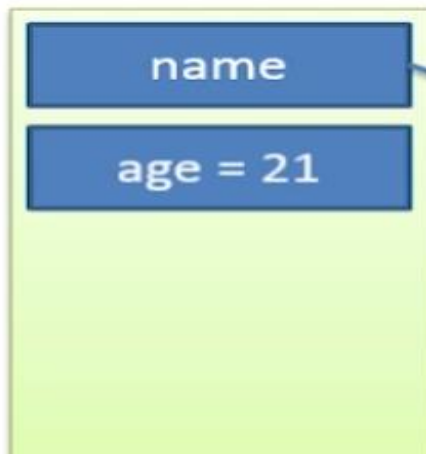




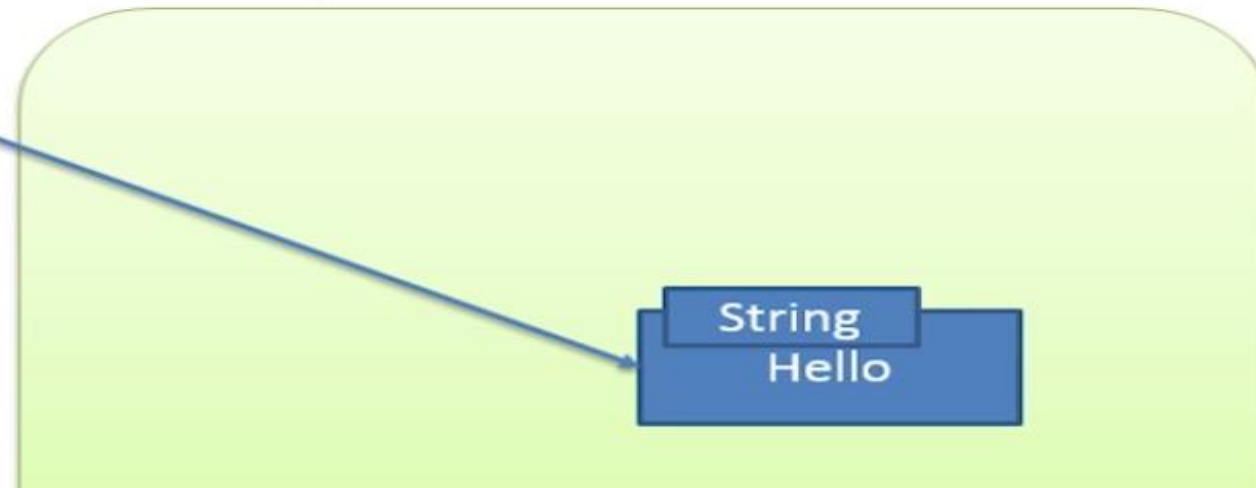
Jak działa pamięć w javie – stereta

```
int age = 21;  
String name = "Hello";
```

Stack



Heap





- Obiekty przechowywane na stercie
- Zmienne to referencje do obiektu
- Zmienne lokalne przechowywane są na stosie

Java memory - start



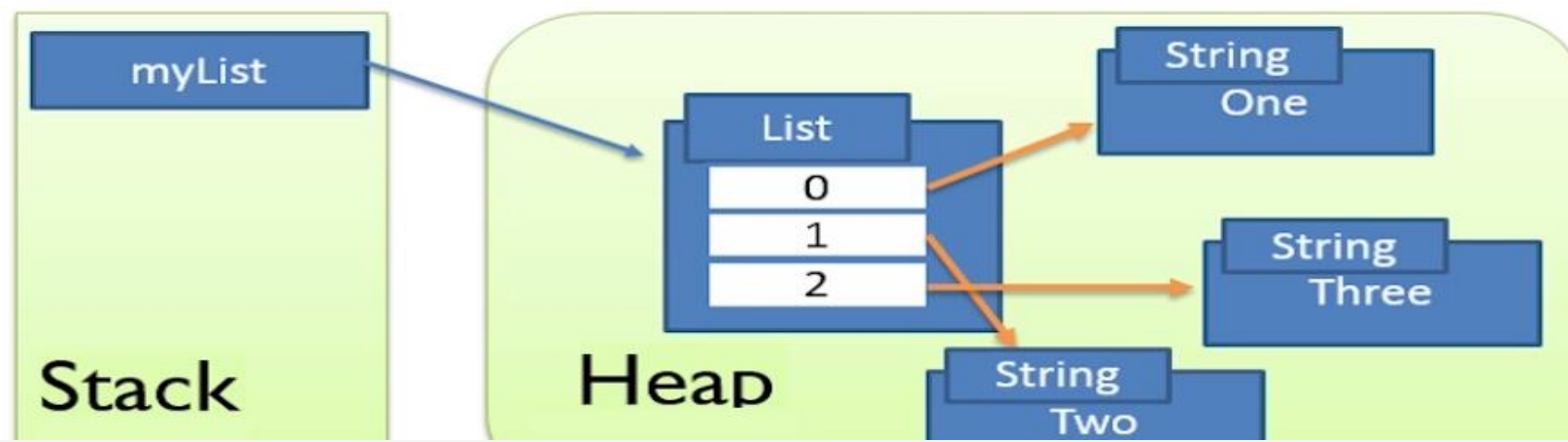
```
public class Main
{
    public static void main(String[] args)
    {
        List<String> myList =
            new ArrayList<String>();
        myList.add("One");
        myList.add("Two");
        myList.add("Three");
        printList(myList);
    }

    public static void printList(List<String> data)
    {
        System.out.println(data);
    }
}
```

Java memory - start



```
public static void main(String[] args) {  
    List<String> myList = new ArrayList<String>();  
    myList.add("One");  
    myList.add("Two");  
    myList.add("Three");  
    printList(myList);  
}  
  
public static void printList(List<String> data) {  
    System.out.println(data);  
}
```



Question - Przekazywanie zmiennych



- Java przekazuje zmienne przez wartość czy referencje?
- Słowo kluczowe final w odniesieniu do zmiennej

Zadanie 5 IMMUTABILITY



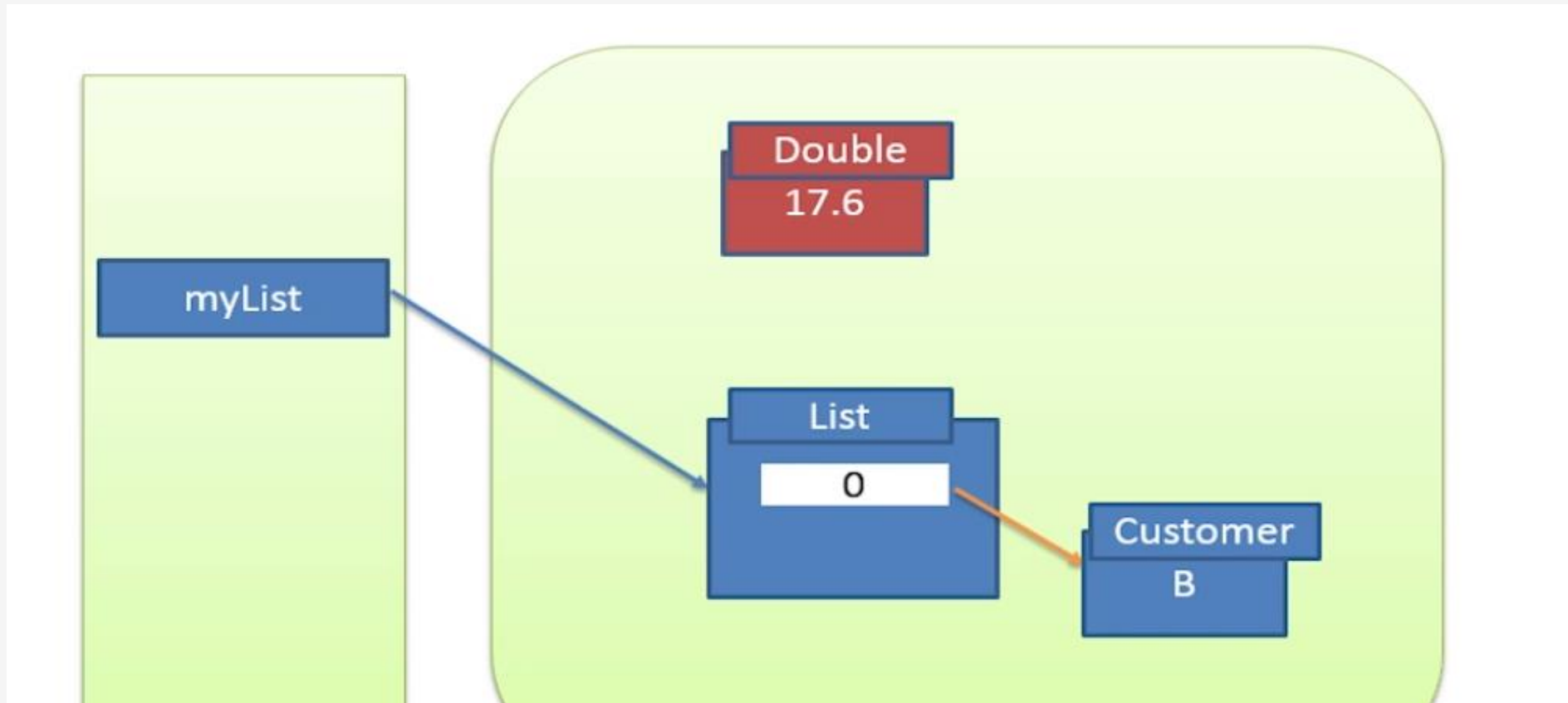
- W projekcie jvm utwórz package immutability
- Potrzebne będą 3 klasy : Book, BookCollection, Price oraz Main
- Klasa Book zawiera pola id, title, author, price (typu Price)
- Klasa Price zawiera pole value, oraz currency
- Klasa BookCollection zawiera 2 metody:
 - findBookByTitle
 - printAllBooks

Zaprojektuj strukturę klas oraz implementacje w/w metod



- W projekcie jvm odszukaj klasy **JavaStack**
- Uruchom prosty program(prawy przycisk i ,Run **JavaStack**)
- Zlokalizuj PID Procesu JVM za pomocą polecenia `jps -l`
 - (lista parametrów <https://docs.oracle.com/javase/7/docs/technotes/tools/share/jps.html>)
- Uruchom narzędzie `jvisualvm.exe`, znajdujące się katalogu `JDK/bin`
- Na podstawie PID uzyskanego wcześniej, sprawdź zawartość sterty

Garbage collector – kiedy zmienna nadaje się do usunięcia

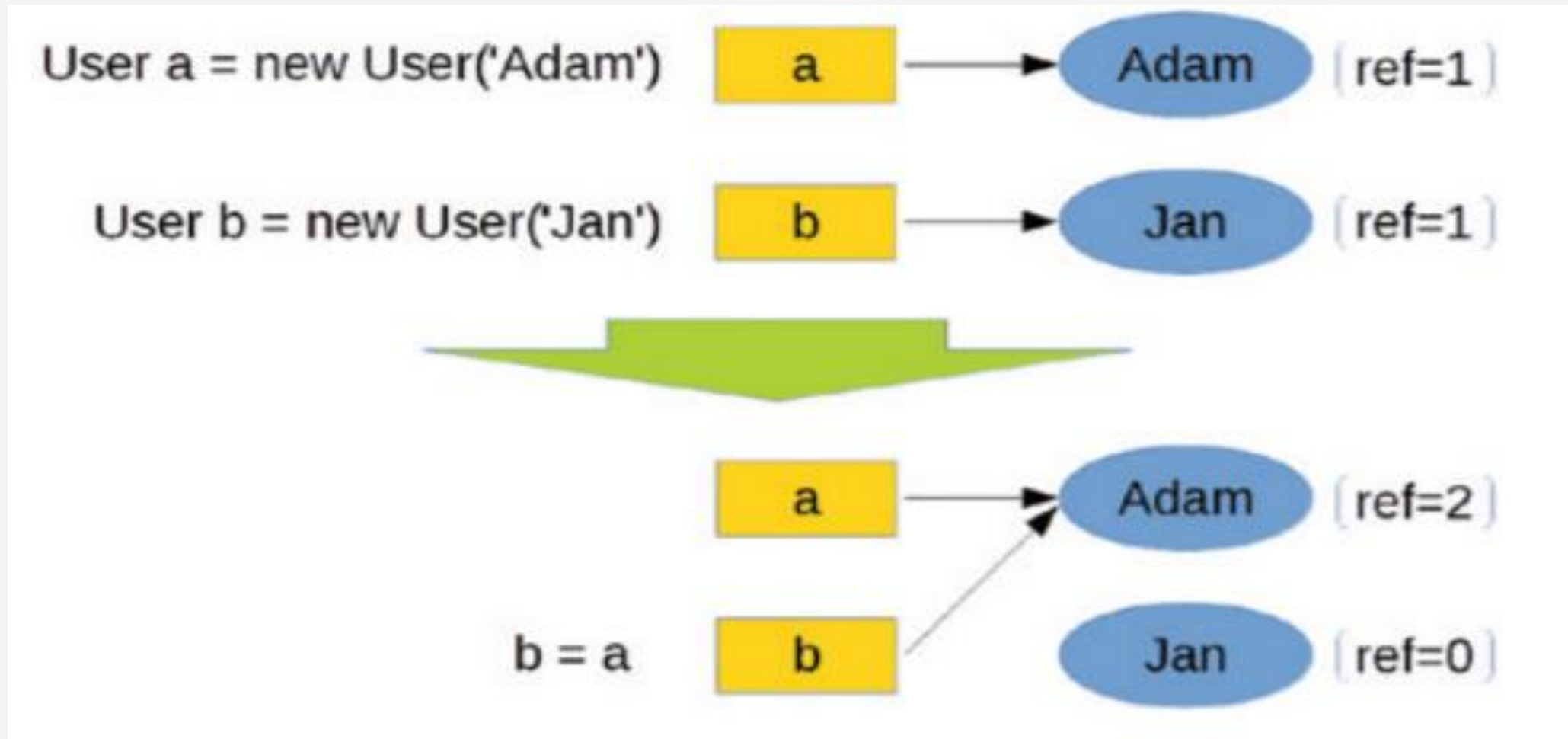


Autor: Mateusz Duraj

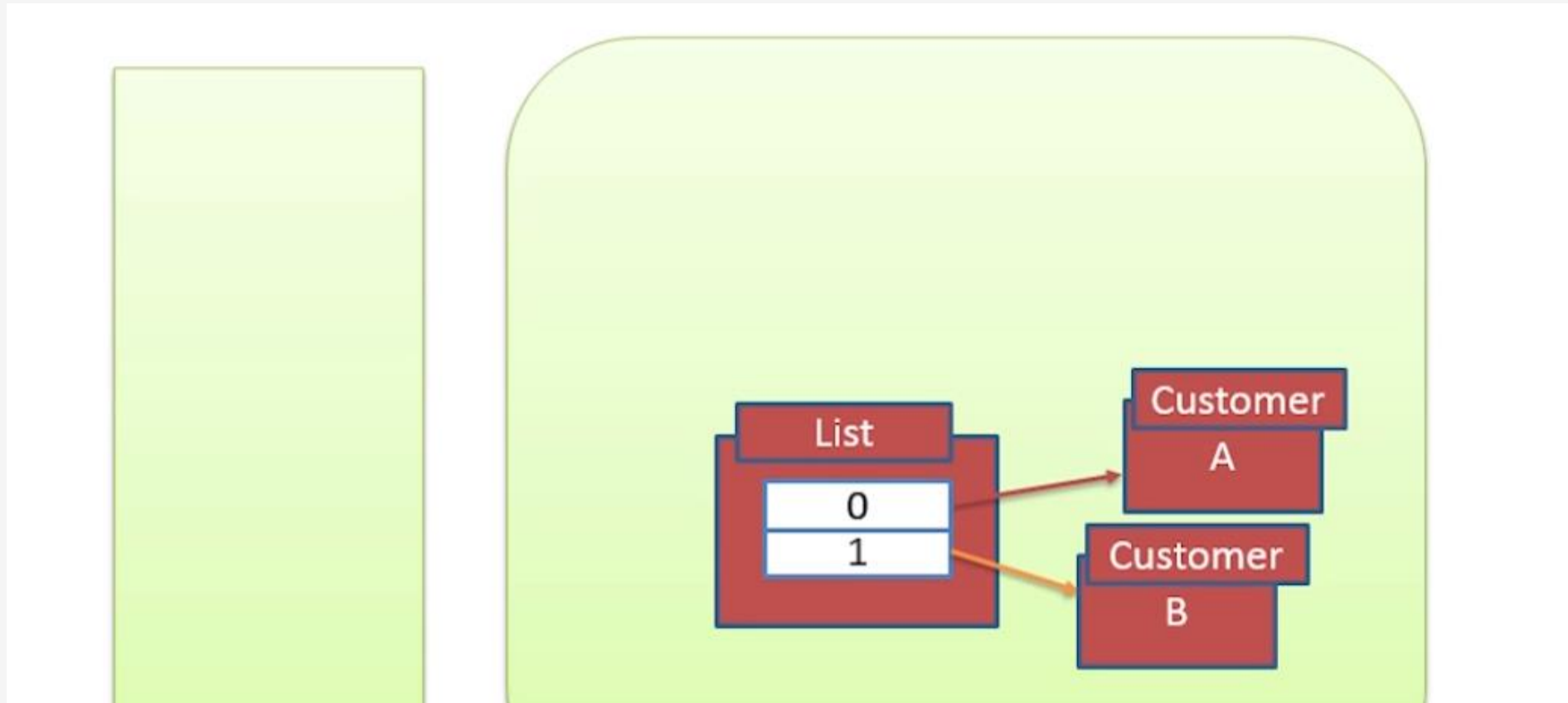
Prawa do korzystania z materiałów posiada Software Development Academy



Garbage collector – kiedy zmienna nadaje się do usunięcia



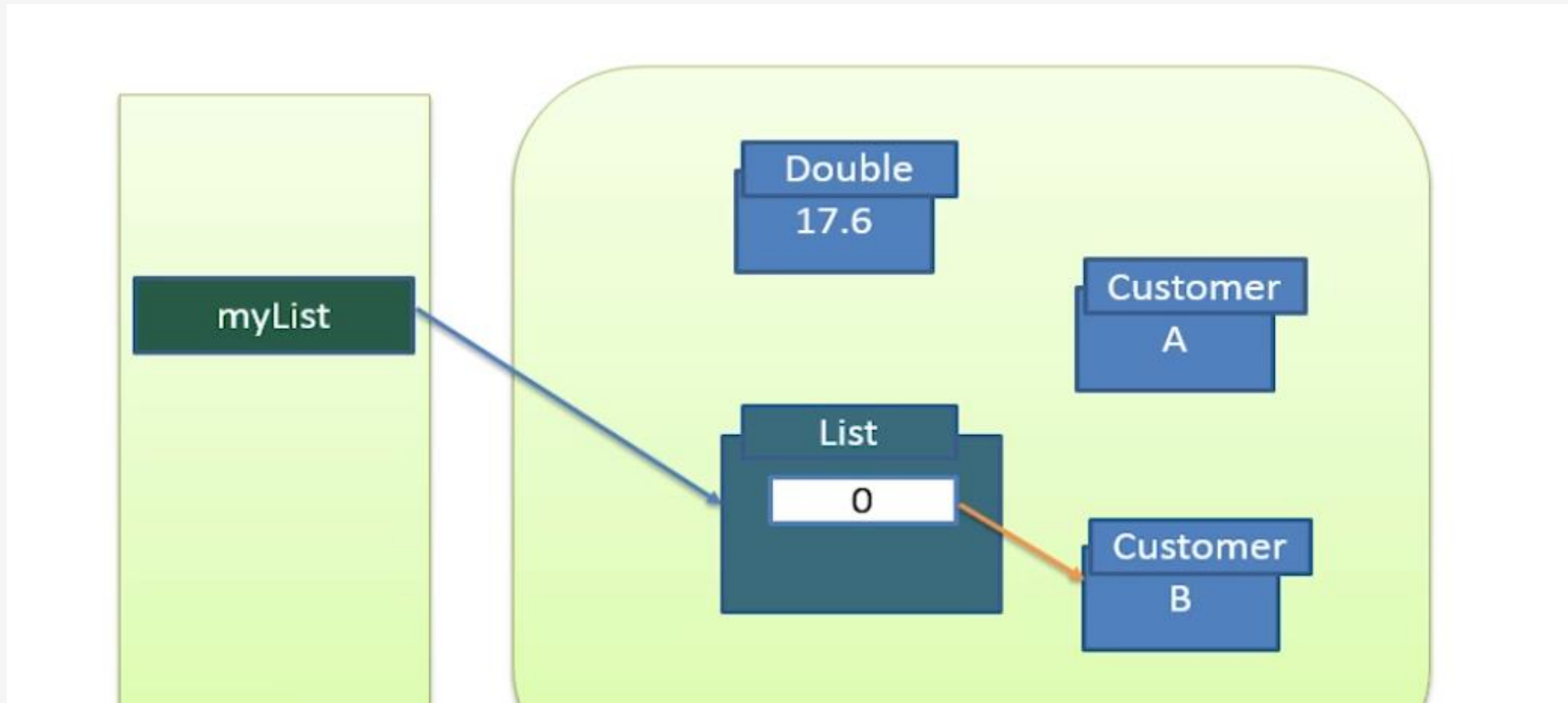
Garbage collector – kiedy zmienna nadaje się do usunięcia



Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy

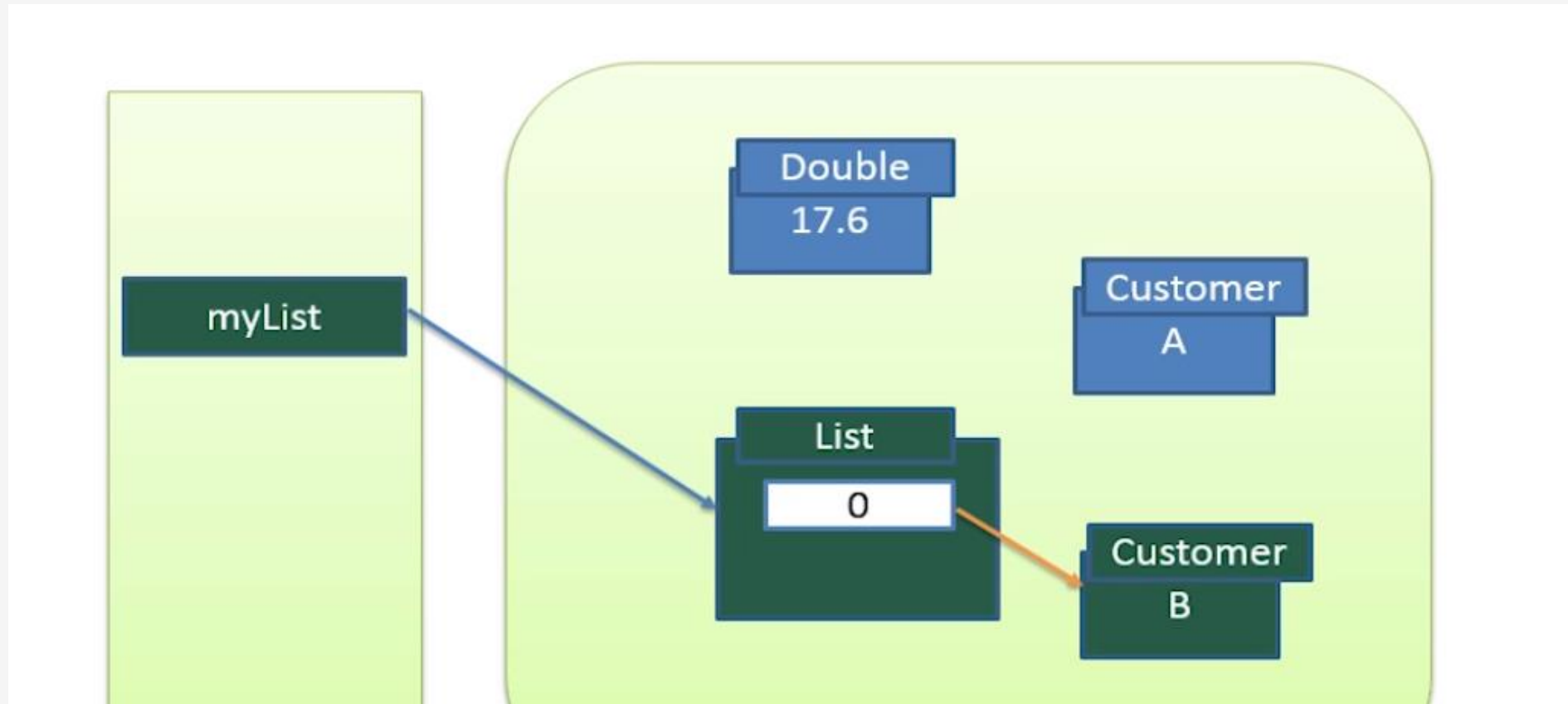
Garbage collector – algorytm "Mark and Sweep"



Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy

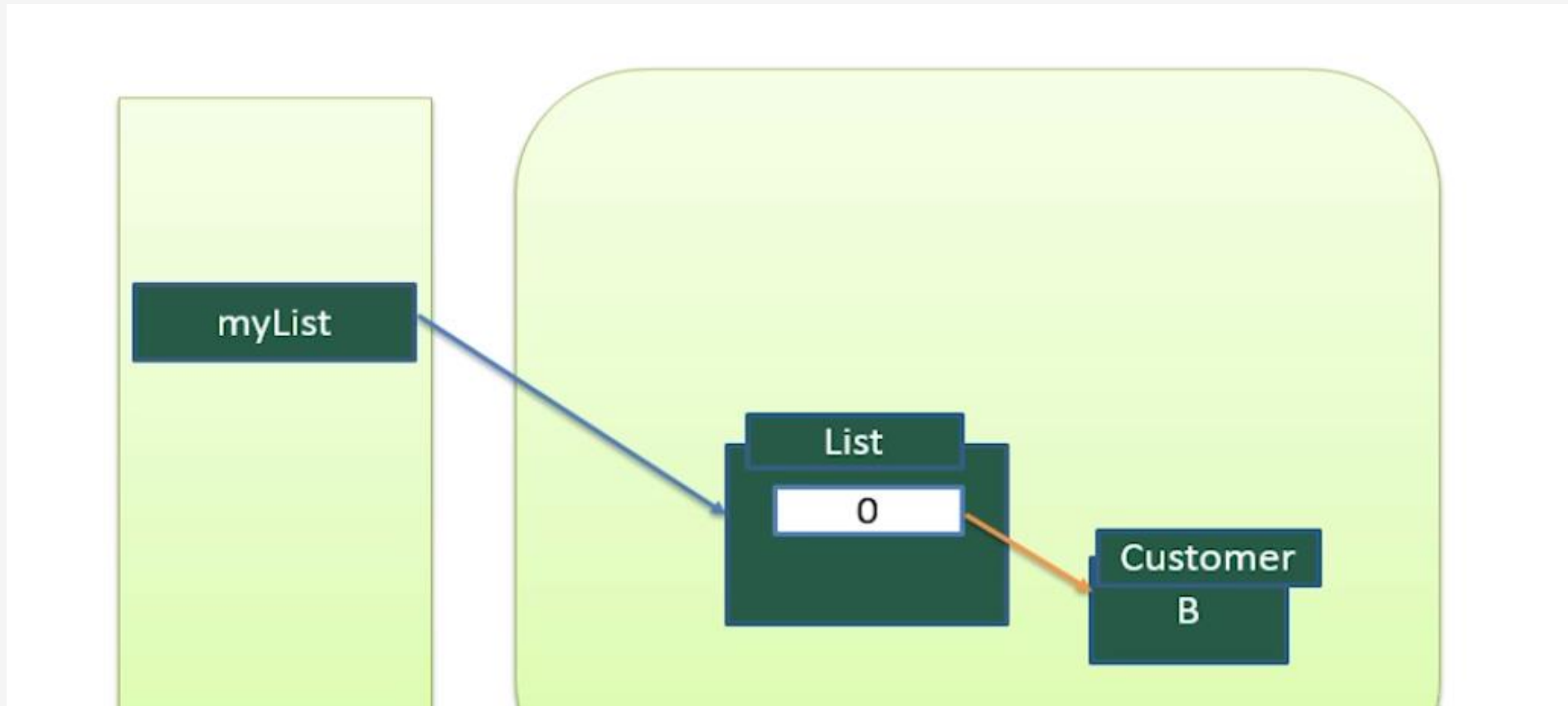
Garbage collector – algorytm "Mark and Sweep"



Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy

Garbage collector – algorytm "Mark and Sweep"

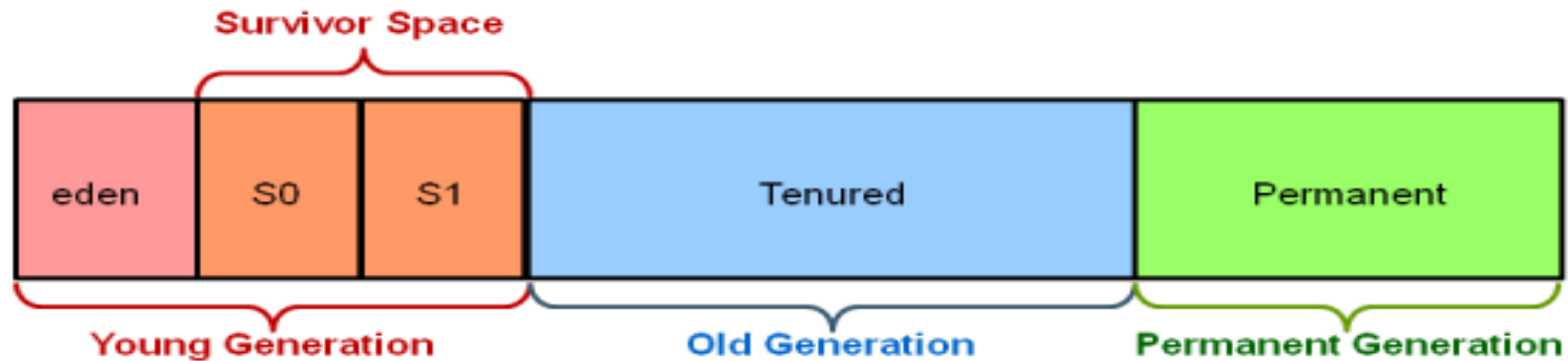


Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy



Hotspot Heap Structure



źródło:

<http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html>



Garbage collector podział sterty

Cała sterta (heap) podzielona jest na dwa obszary:

- Młodą (Young) generację
- Starą (Old / tenured) generację

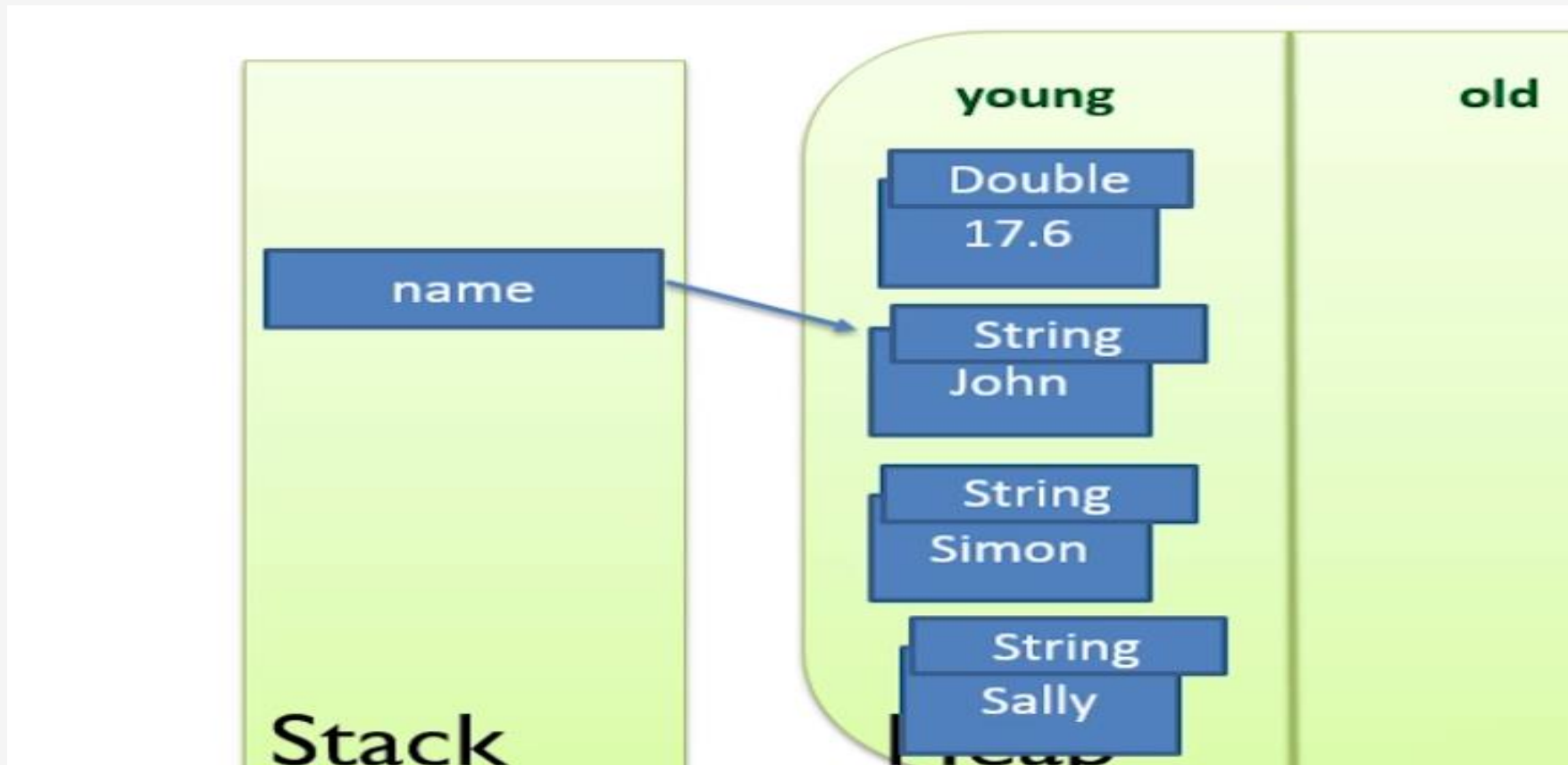
Związku z czym proces gc(garbage collector) podzielony jest na 3 fazy:

- małą kolekcję (minor GC) – obejmującą młodą generację,
- dużą kolekcję (major GC) – polegającą na sprzątaniu starej generacji
- pełną kolekcję (full GC) – obejmującą całą stertę.

Garbage collector



- Większość obiektów w Javie żyje przez bardzo krótki czas



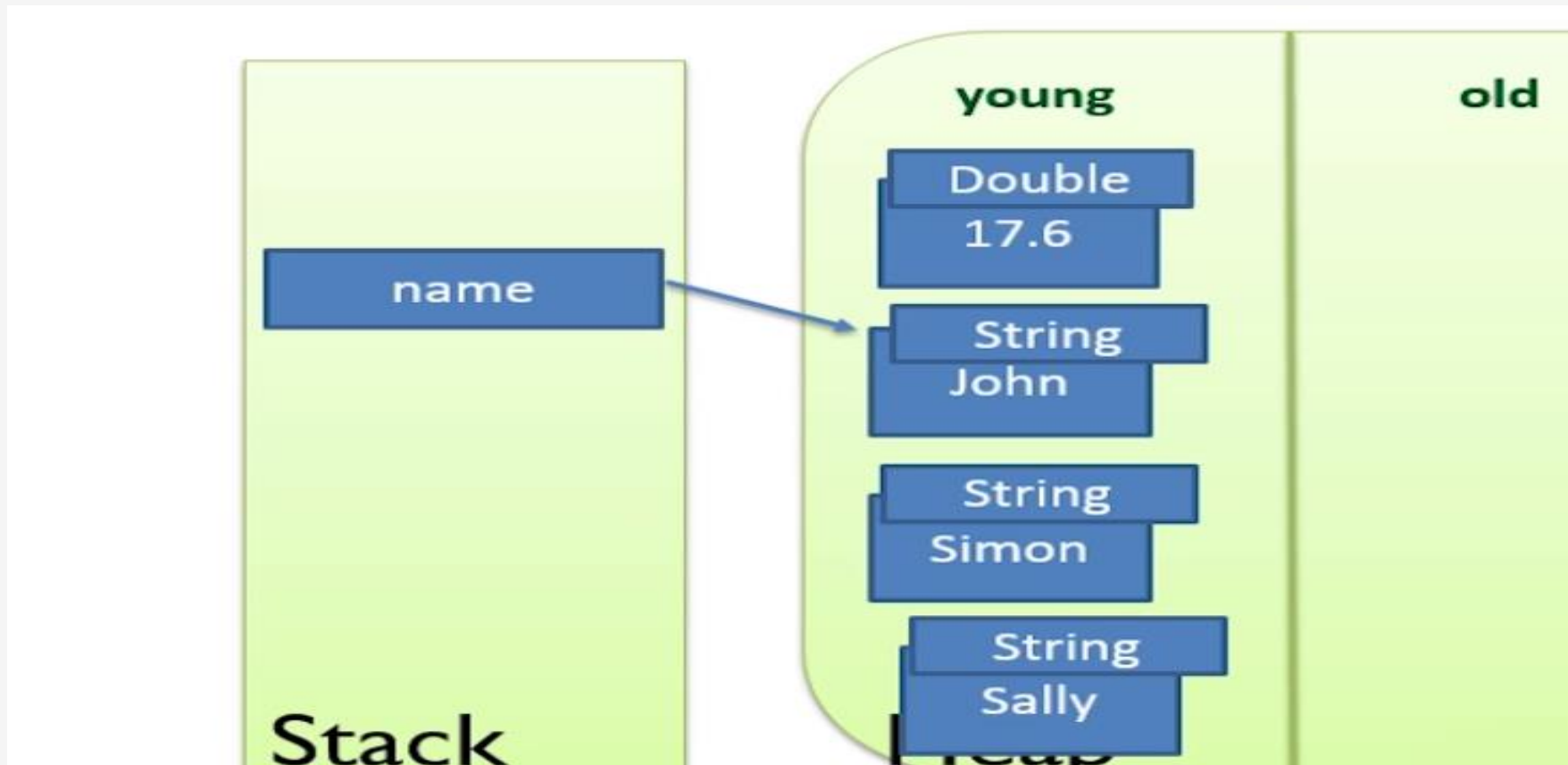
Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy

Garbage collector



- Większość obiektów w Javie żyje przez bardzo krótki czas



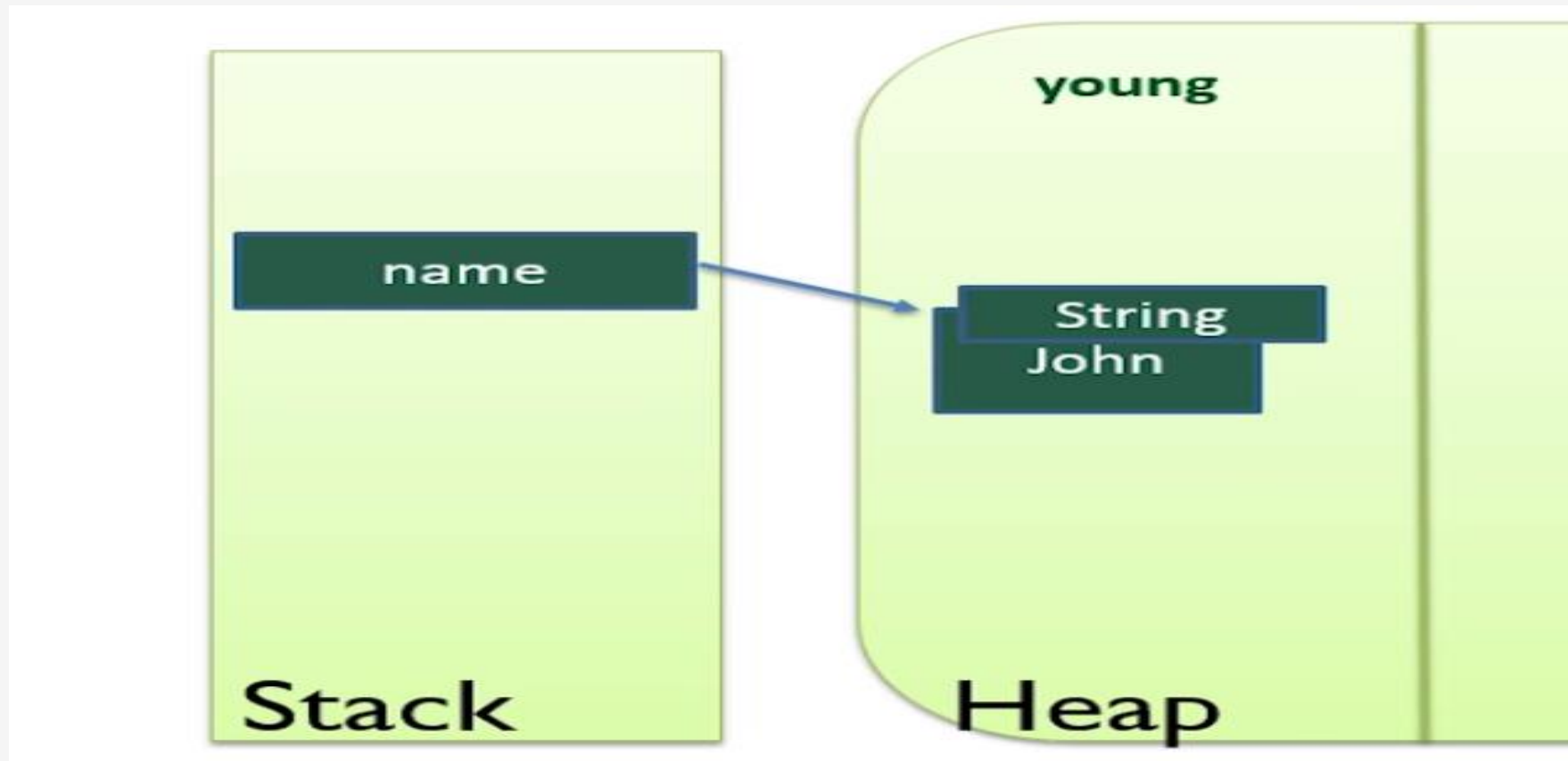
Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy

Garbage collector



Mark and sweep – na stercie pozostaje jeden obiekt



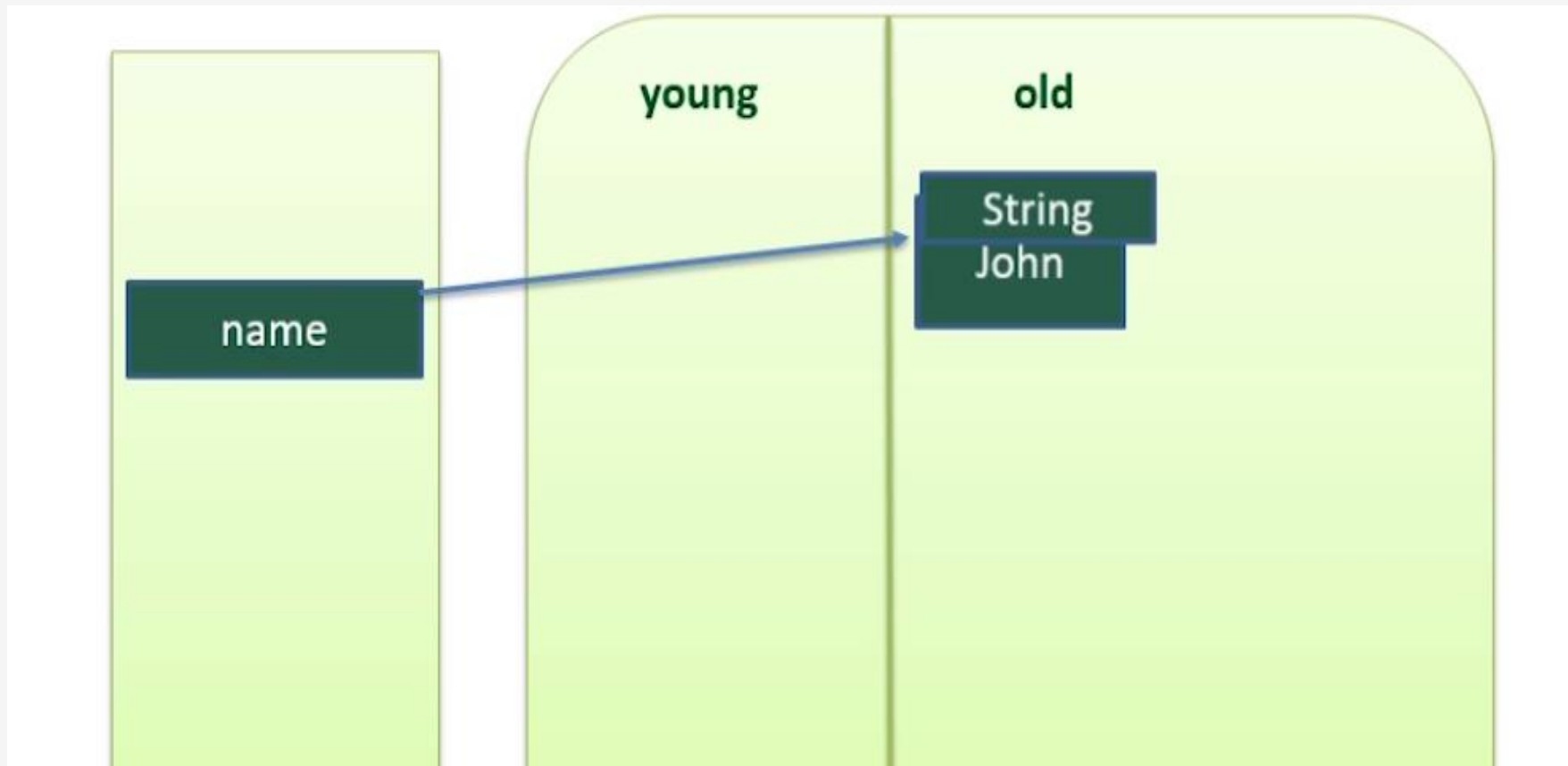
Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy

Garbage collector



Obiekt String "awansował" do old generation



Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy



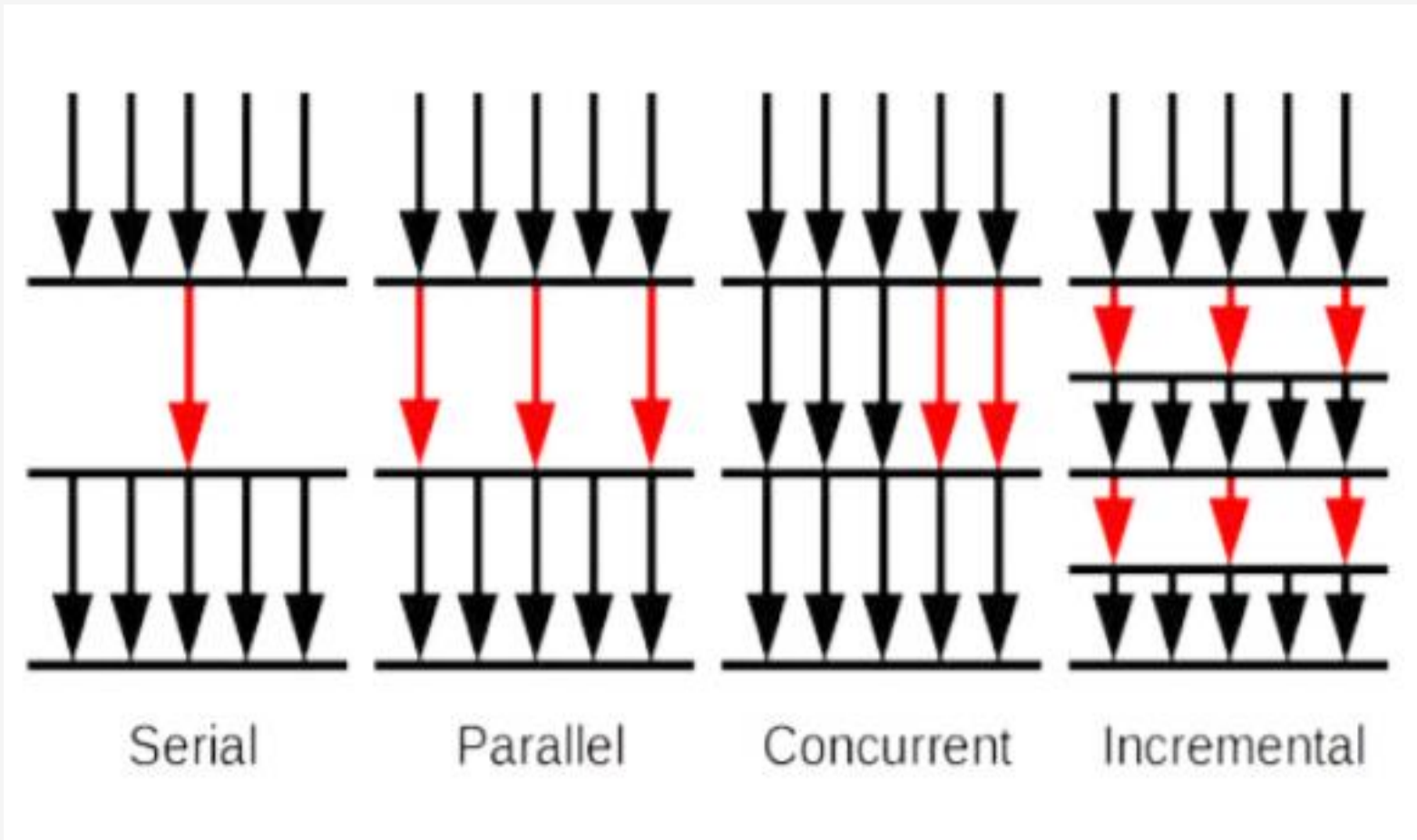
Garbage collector - tuning

- `-Xms` – max heap size
- `-Xms` starting heap size
- `-XX:MaxPermSize` set size of the permgen
- `-verbose:gc` – print to the console when a garbage collection take place
- `-Xmn` set the size of the young generation
- `-XX:HeapDumpOnOutOfMemory` – creates a heap dump file



- Serial – gdy wątek GC startuje, aplikacja jest zatrzymywana, dobry dla małych aplikacji, wybór –XX: +UseSerialGC
 - Parallel – osobny wątek gc – gdy mamy kilka CPU, dobry dla dużych aplikacji –XX: UseParallelGC
 - The CMS Collector – dla aplikacji >100MB, z wieloma procesorami, większość wykonania gc / praca aplikacji współbieżnie
 - The G1 Collector – zalecany dla serwerów, do obsługi powyżej >4GB
-
- `java -XX:+PrintCommandLineFlags -version` < wypisuje defaultowy gc

Garbage collector - algorytmy



Autor: Mateusz Duraj

Prawa do korzystania z materiałów posiada Software Development Academy



Stack vs Heap – różnice ?



Stack trace

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
```

```
Exception in thread "main" java.lang.NullPointerException: Cannot be null  
    at pl.sdacademy.jvm.jit.JitTrain.sayHello(JitTrain.java:20)  
    at pl.sdacademy.jvm.jit.JitTrain.testLoopUnrolling(JitTrain.java:13)  
    at pl.sdacademy.jvm.jit.JitTrain.main(JitTrain.java:7)
```

```
Process finished with exit code 1
```



Debugging



A screenshot of the IntelliJ IDEA IDE's 'Frames' window. The window title is 'Frames'. Below the title bar, there is a header bar with a red checkmark icon, the text '"main"@1 in group "main": RUNNING', a dropdown arrow, and three navigation icons (up, down, and a funnel). The main area of the window contains a list of frames: 'sayHello:20, JitTrain (pLsdacademy.jvm.jit)' (highlighted in blue), 'testLoopUnroling:13, JitTrain (pLsdacademy.jvm.jit)', and 'main:7, JitTrain (pLsdacademy.jvm.jit)'. A context menu is open over the 'sayHello' frame, displaying the following options: 'Drop Frame' (with a red 'x' icon), 'Throw Exception', 'Force Return', 'Copy Stack' (with a document icon), 'Export Threads...' (with a magnifying glass icon), 'Customize Threads View...', 'Add Stepping Filter...', and 'Async Stack Traces' (with a checkmark icon).



W Javie wycieki powinny nie występować dzięki JVM i zarządzaniu pamięcią przez GC – jednak należy uważać 😊



- JMH służy do tworzenie benchmarków w Javie czyli badania wydajności fragmentów naszej aplikacji.
- <http://hg.openjdk.java.net/code-tools/jmh/file/tip/jmh-samples/src/main/java/org/openjdk/jmh/samples/> - przykłady użycia



Narzędzia – pozwalające na pracę z JVM

- **Javap** – generuje czytelny byte code
- **Jstack** – zrzuty wątków
- **Visual vm** – podpinamy się do działającej aplikacji, i sprawdzamy tzn. telemetry
- **Mission control** – podpinamy się do działającej aplikacji, i sprawdzamy tzn. metryki (np. zajętość pamięci)
- **Jprofiler** – płatny, kombajn do analizy aplikacji, bardzo dokładna np. gdzie są tworzone obiekty, graf obiektów, ile duplikowanych stringów, największe obiekty. Polecam wykorzystać trial jeśli chcecie sprawdzić jak działa java
- **Java object layout** – drukuje jak wygląda layout klasy, np. ile zajmują pamięci klasy, różnice pomiędzy boolean a Boolean.



JDK do pobrania jako projekt:

<http://hg.openjdk.java.net/jdk8/jdk8>

<http://tutorials.jenkov.com/java-performance/jmh.html>

<https://programistanaswoim.pl/wp-content/uploads/2017/07/jvm1.pdf>

<https://bottega.com.pl/pdf/materialy/jvm/jvm2.pdf>



<http://java.meritcampus.com>

<https://www.slideshare.net/AlfonsoRuzafaMolina/scala-fundamentals>

<https://www.stechies.com/difference-between-permgen-metaspace/>

<https://www.ibm.com/developerworks/library/j-dclp1/index.html>

<https://blog.takipi.com/java-on-steroids-5-super-useful-jit-optimization-techniques/>



<http://docs.oracle.com/javase/specs/jvms/se8/html/>. - specyfikacja JVM