Just-In-Time compiler - ukryty „przyjaciel”

Krzysztof Ślusarski

Krótko o mnie

- Programuję od 1992

britenet
quality for lease

Krótko o mnie

- Programuję od 1992

- Zawodowo:
  - Od 10.2006 – 08.2007 – RODO
  - Od 09.2007 – Britenet sp z. o. o
    Java Programmer /
    Team Leader /
    System Architect /
    Solution Architect

britenet
quality for lease

Krótko o mnie

- Poza 8h 5/7:
  - Szkolenia

britenet
quality for lease

Krótko o mnie

- Poza 8h 5/7:
  - Szkolenia
  - Diagnoza awarii produkcyjnych + profiling

Krótko o mnie

- Poza 8h 5/7:
  - Szkolenia
  - Diagnoza awarii produkcyjnych + profiling
  - Stolarstwo meblowe

Krótko o mnie

- Poza 8h 5/7:
  - Szkolenia
  - Diagnoza awarii produkcyjnych + profiling
  - Stolarstwo meblowe
- Prywatnie:

britenet
quality for lease

Wyłączenie odpowiedzialności

- Nie tworzę JVM

Wyłączenie odpowiedzialności

- Nie tworzę JVM
- Wszystko co mówię, może być kłamstwem

Wyłączenie odpowiedzialności

- Nie tworzę JVM
- Wszystko co mówię, może być kłamstwem
- Pytania na końcu

Wyłączenie odpowiedzialności

- Nie tworzę JVM
- Wszystko co mówię, może być kłamstwem
- Pytania na końcu
- W 1,5h się nie wyrobię

Motto

Nie ma w życiu nic za darmo

britenet
quality for lease

Wstęp

- Konsekwencje

britenet
quality for lease

Wstęp

- Konsekwencje
- Słowniczek

britenet
quality for lease

Wstęp

- Konsekwencje
- Słowniczek
  - Local safepoint

Wstęp

- Konsekwencje
- Słowniczek
  - Local safepoint
  - Global safepoint

Wstęp

- Konsekwencje
- Słowniczek
  - Local safepoint
  - Global safepoint
  - Safepoint opperation

Wstęp

- Konsekwencje
- Słowniczek
  - Local safepoint
  - Global safepoint
  - Safepoint opperation
  - Stop the world

Wstęp

- JIT vs AOT

Wstęp

- JIT vs AOT
- Zależy od implementacji JVM – Oracle Hotspot

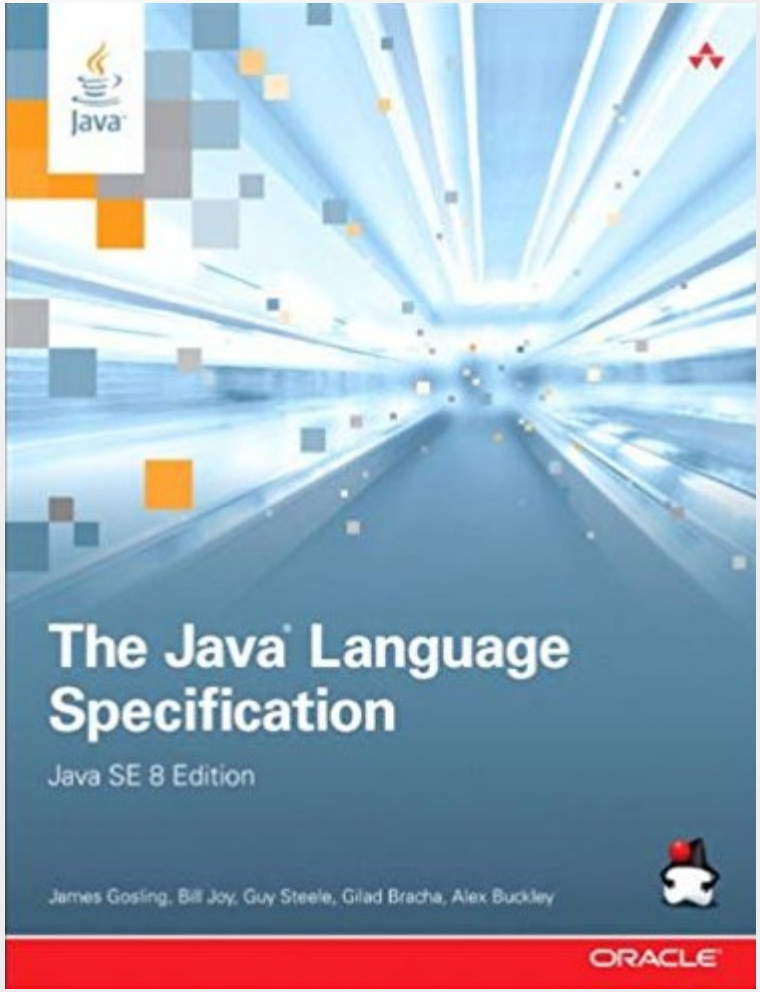
Wstęp

- JIT vs AOT
- Zależy od implementacji JVM – Oracle Hotspot
- Optymalizuje:
  - Metody – ivocation counter
  - Pętle – backedge counter, OSR

Wstęp

- JIT vs AOT
- Zależy od implementacji JVM – Oracle Hotspot
- Optymalizuje:
  - Metody – ivocation counter
  - Pętle – backedge counter, OSR
- Co nie jest zabronione jest dozwolone

Wstęp

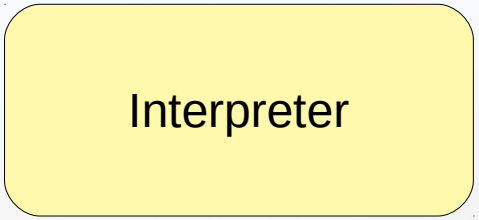


britenet
quality for lease

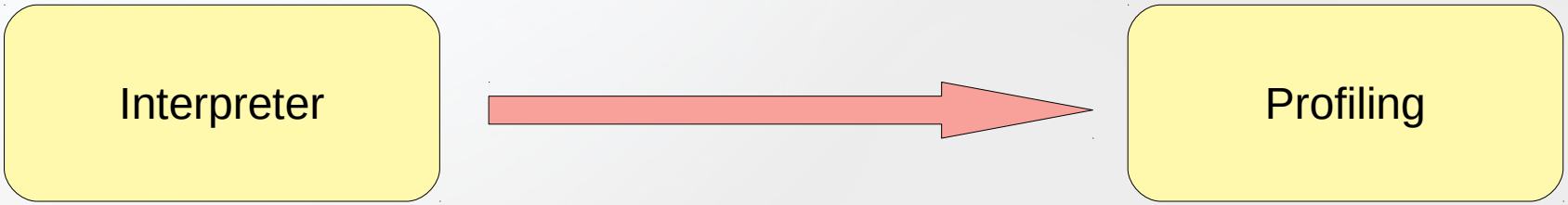
Cykl życia

britenet
quality for lease

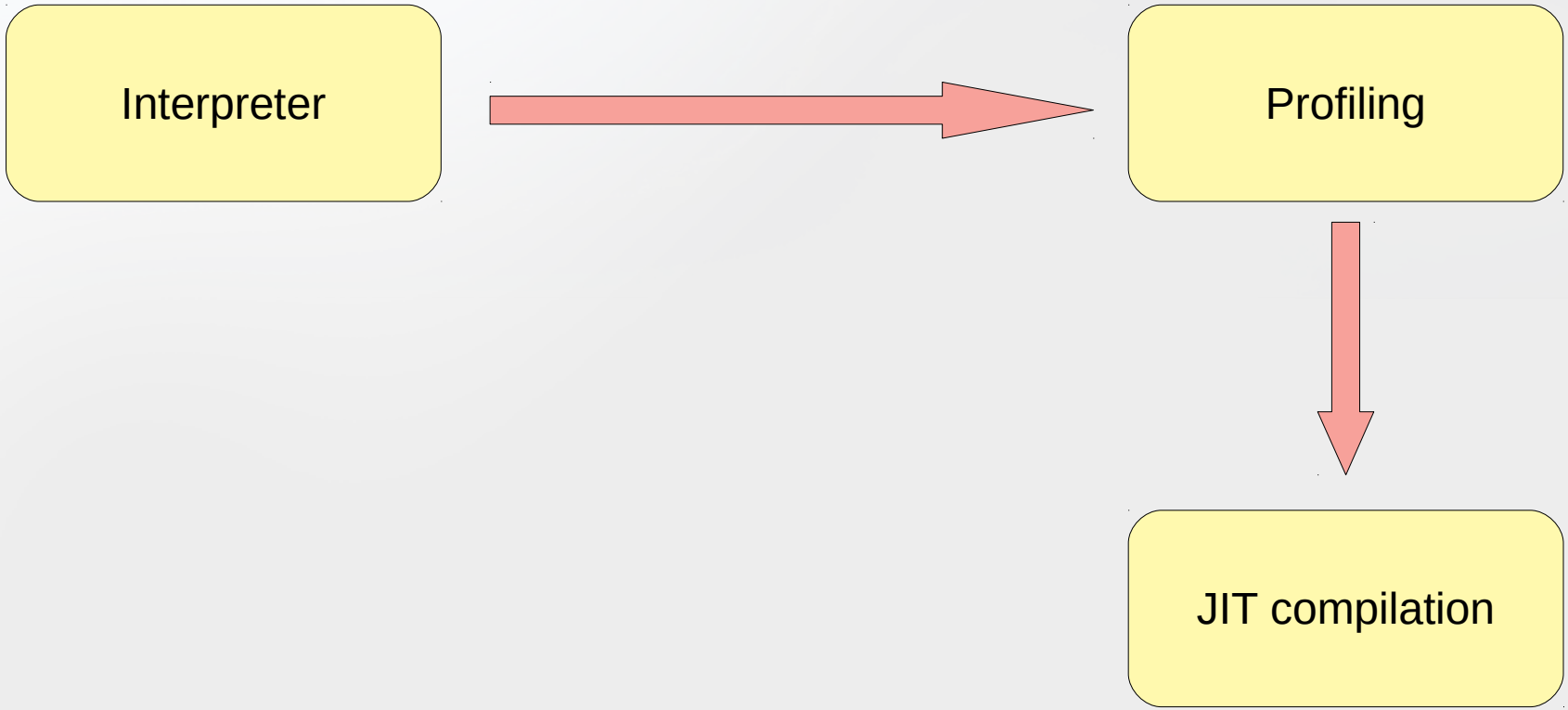
Cykl życia



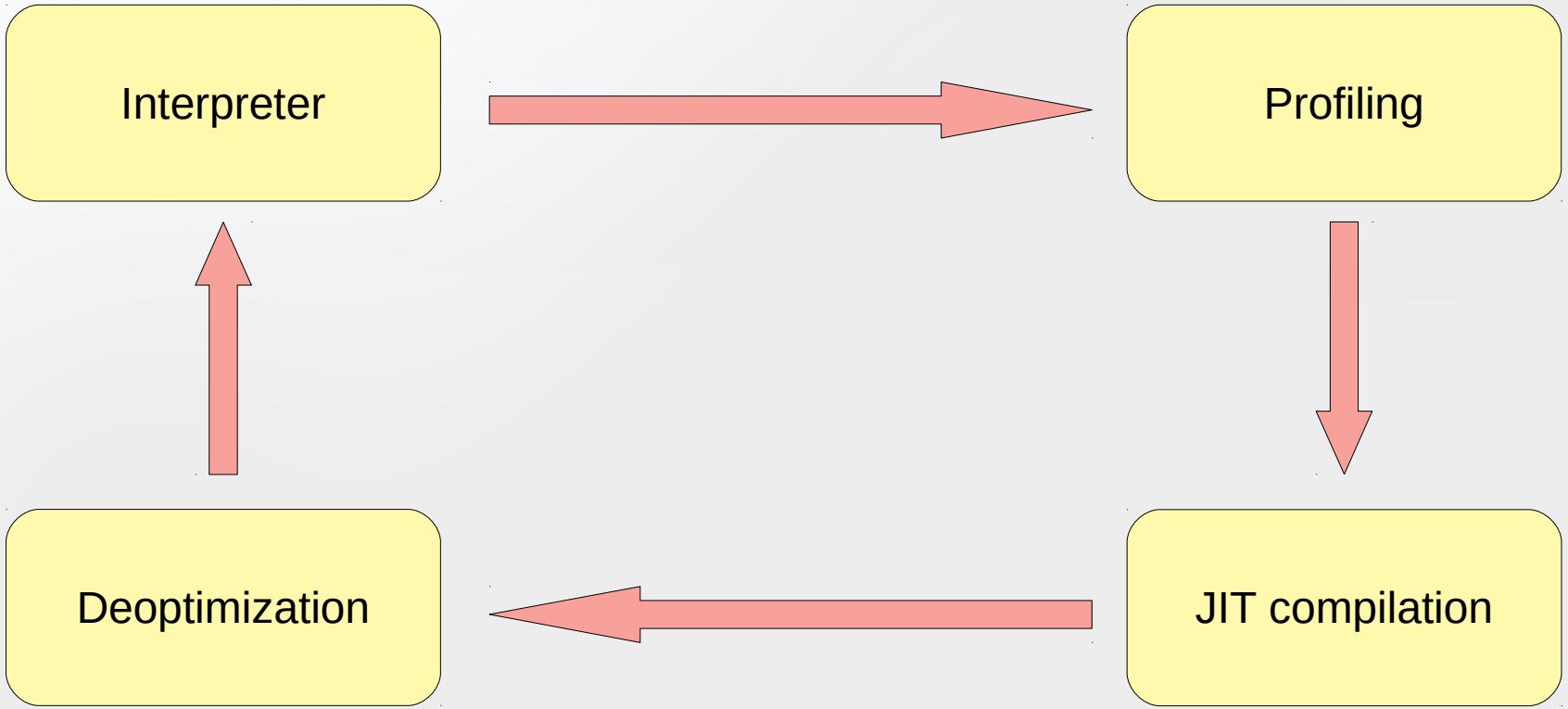
Cykl życia



Cykl życia



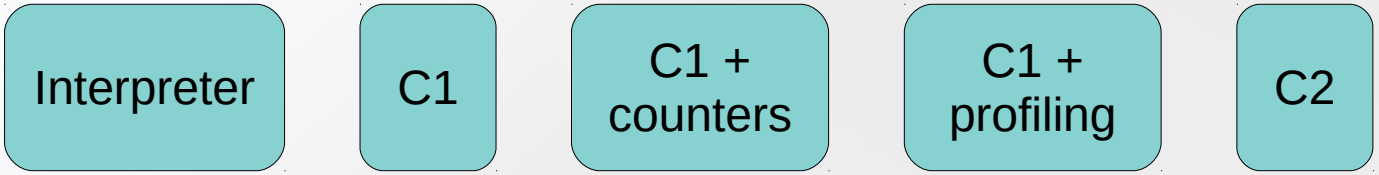
Cykl życia



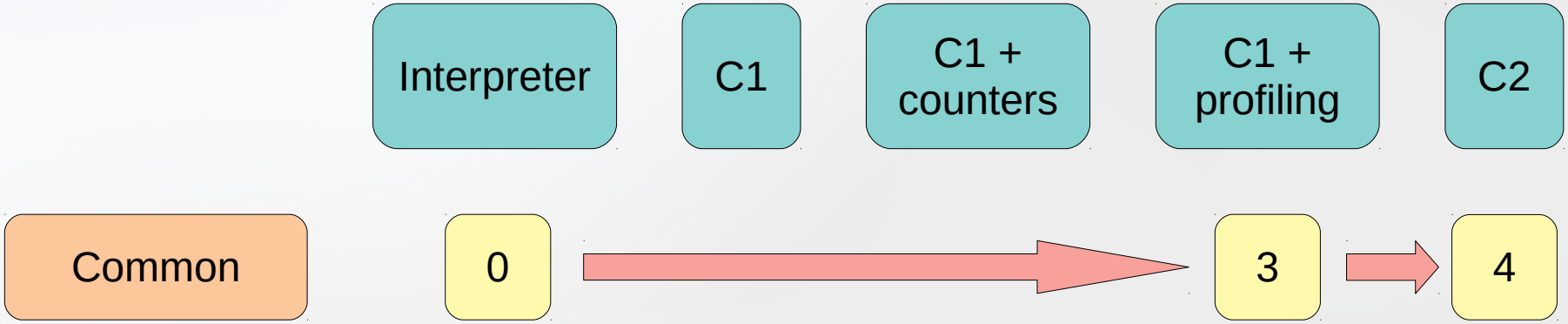
Kompilatory

- C1 – client – 5 razy szybszy
- C2 – server – 10-100+ razy szybszy

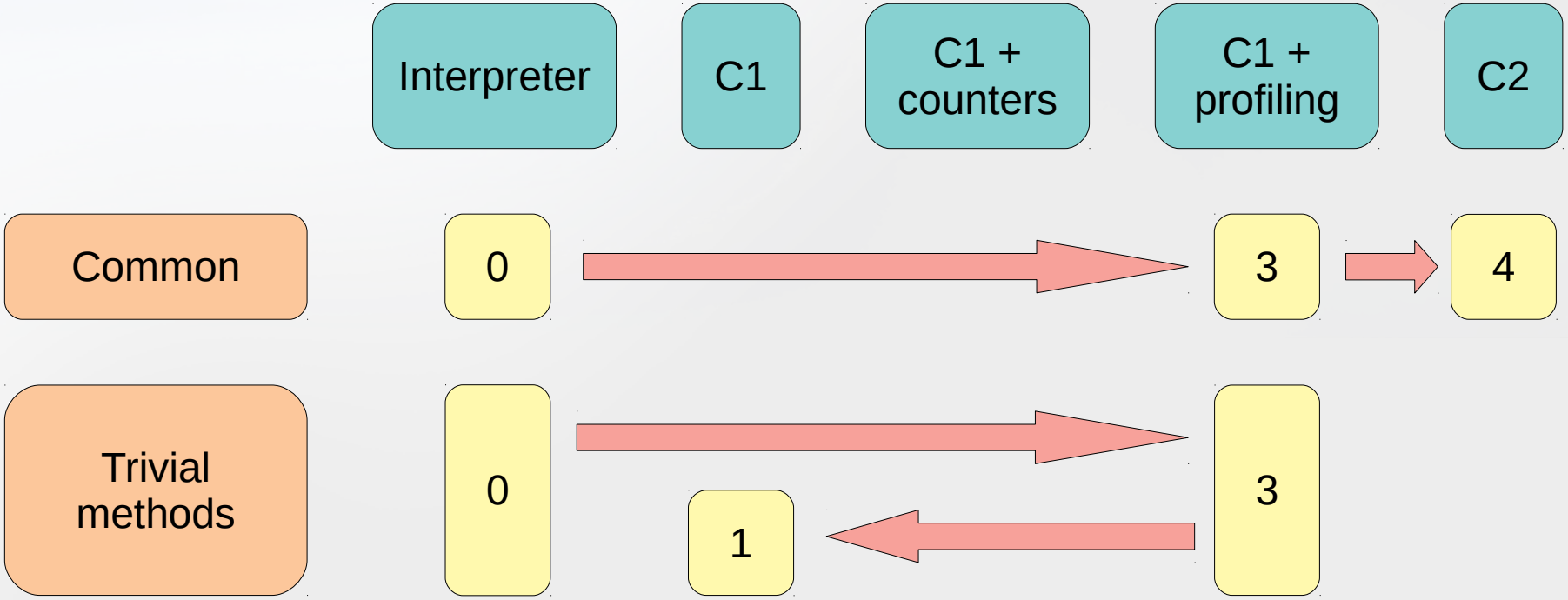
Tiered compilation



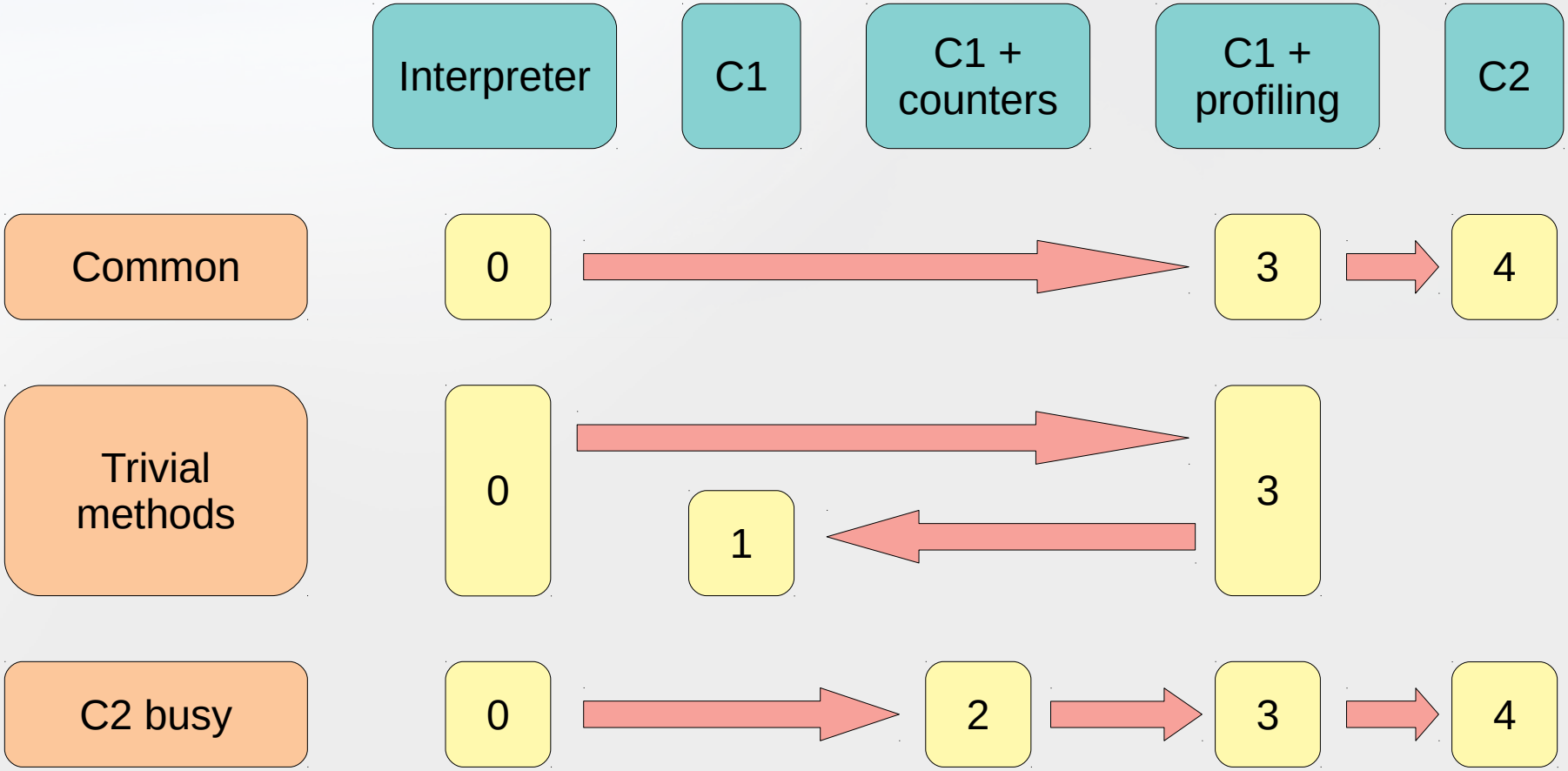
Tiered compilation



Tiered compilation



Tiered compilation



-XX:+PrintCompilation

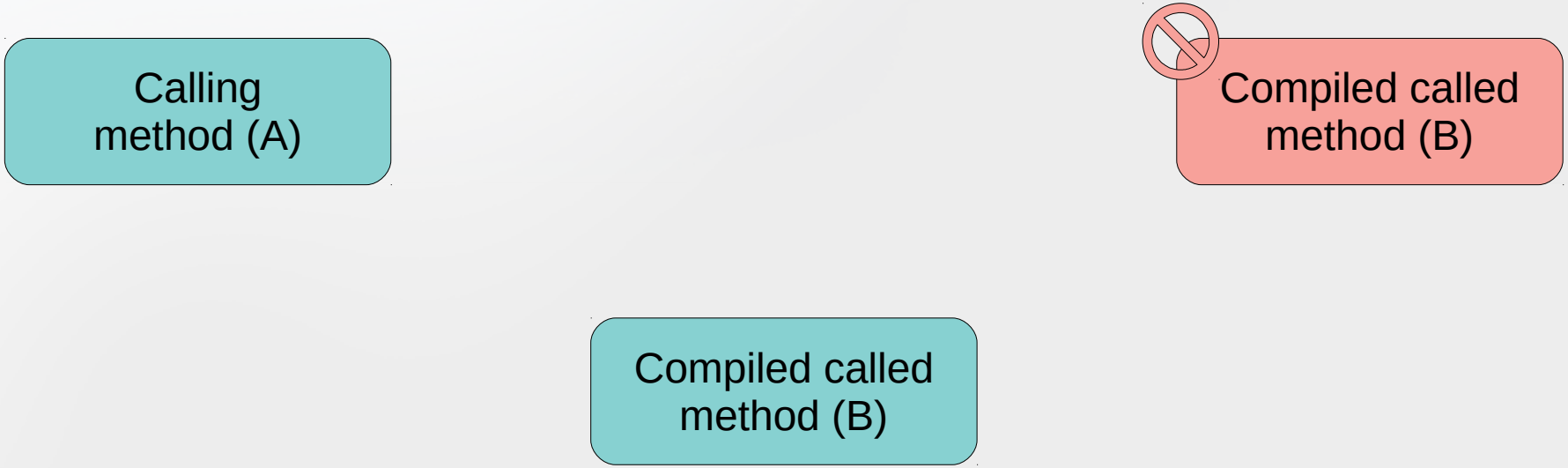
```
   50   37      n 0       java.lang.Object::hashCode (native)
   08  167    !   4       java.lang.ClassLoader::loadClass (122 bytes)
  159  376   s    3       java.io.ByteArrayOutputStream::write (32 bytes)
  240  388 %      3       bt.AOptCls::run @ 8 (39 bytes)
  309  405         3       bt.AOptCls::doSomething (29 bytes)   made not entrant
16660  955         2       com.sun.proxy.$Proxy3::run (27 bytes)   made zombie
```

„n” - native
„!” - exception handler
„s” - synchronized
„%” - OSR

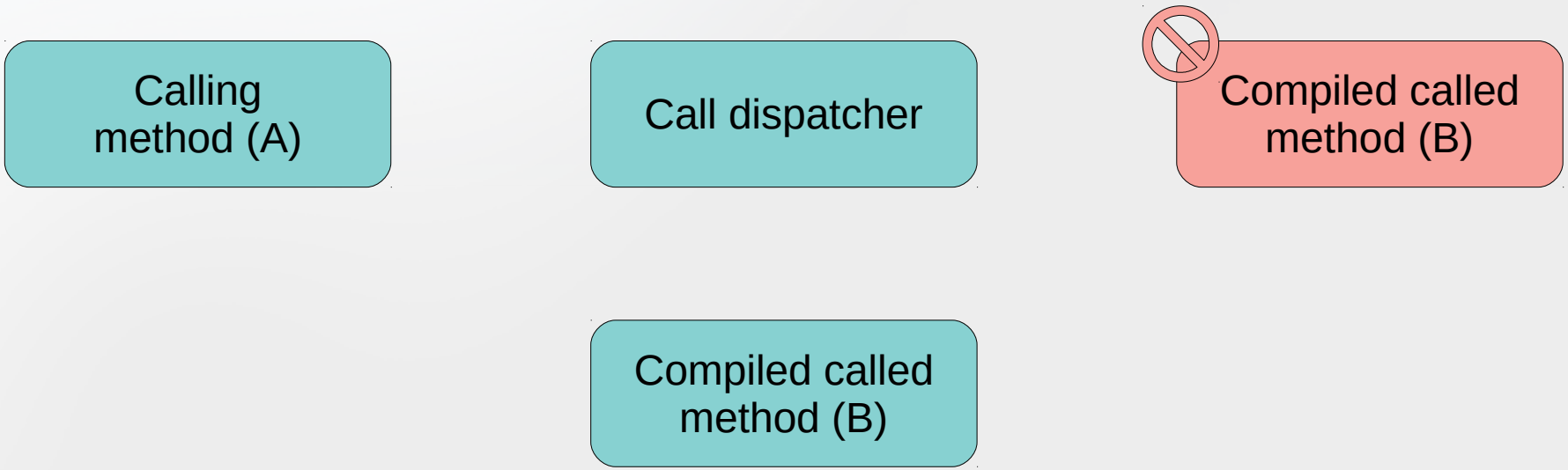
Timestamp / ID kompilacji / Tier / Rozmiar metody / Indeks bytecodu pętli / Info

britenet
quality for lease

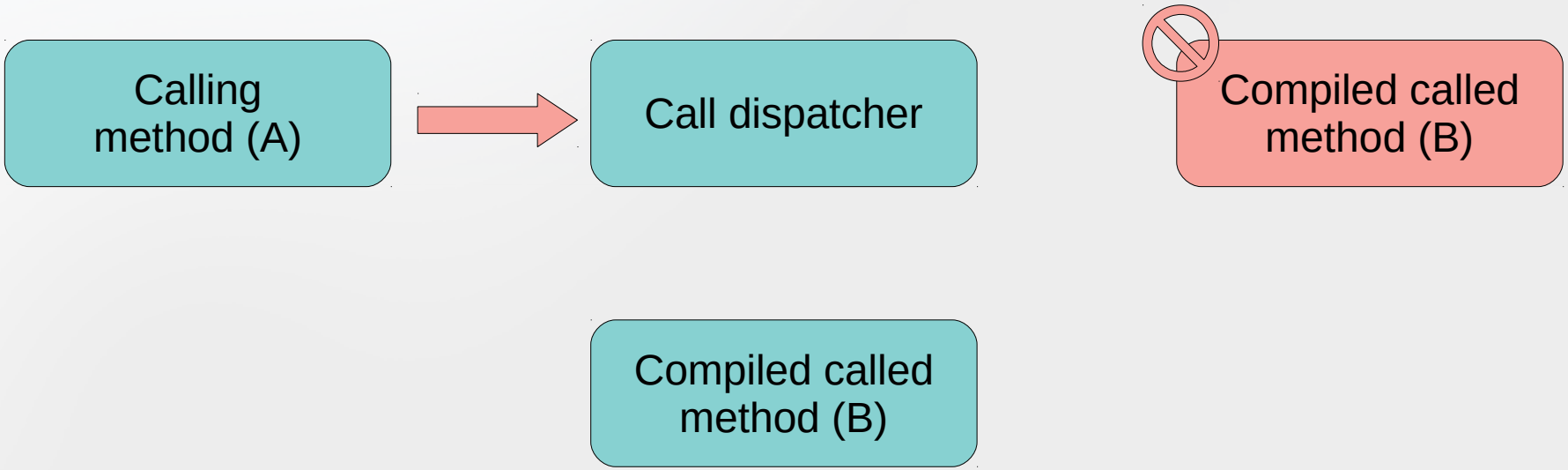
Made not entrant



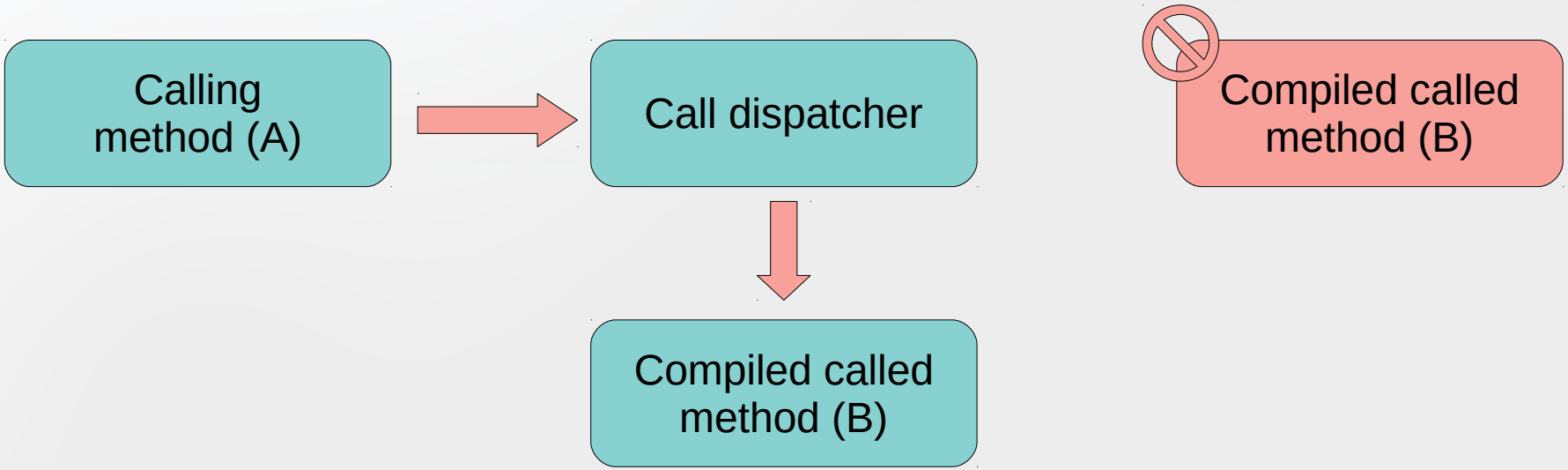
Made not entrant



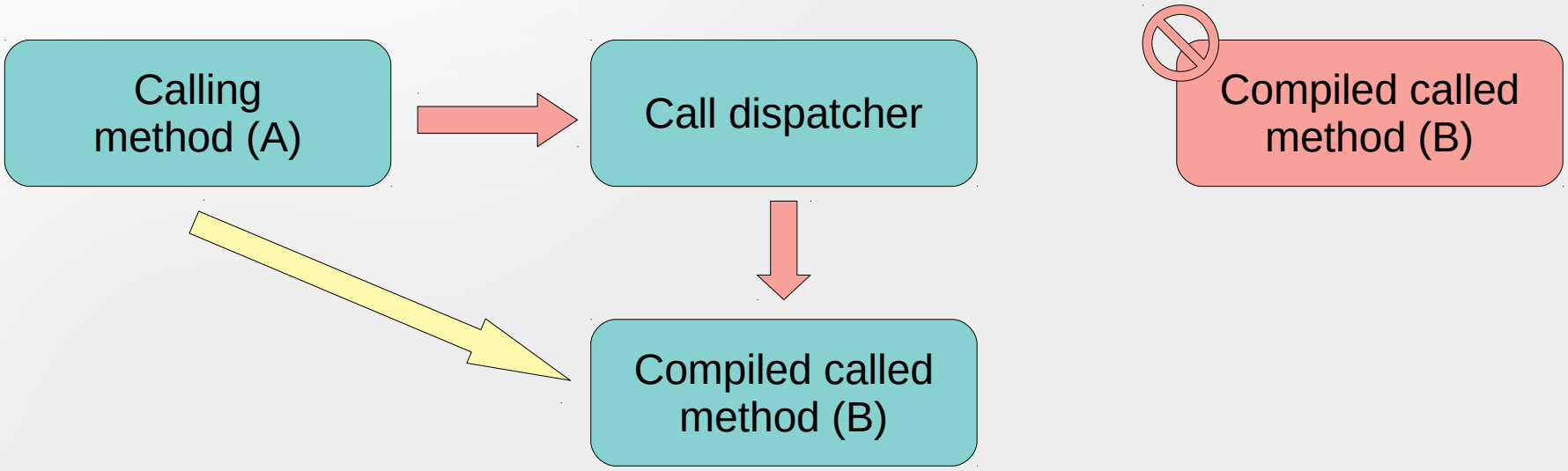
Made not entrant



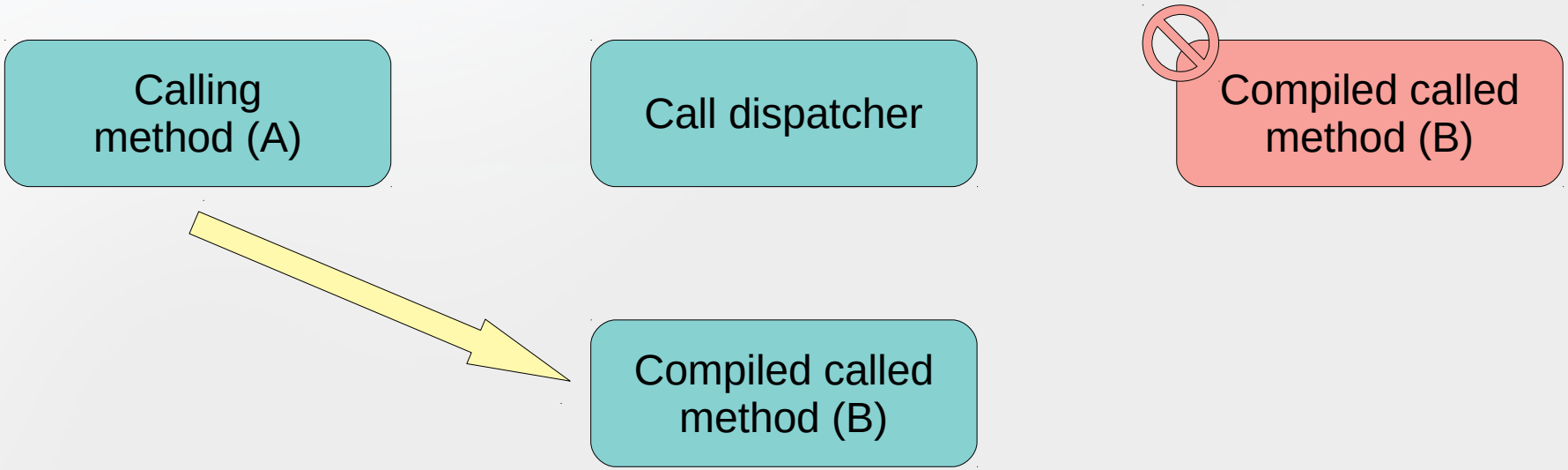
Made not entrant



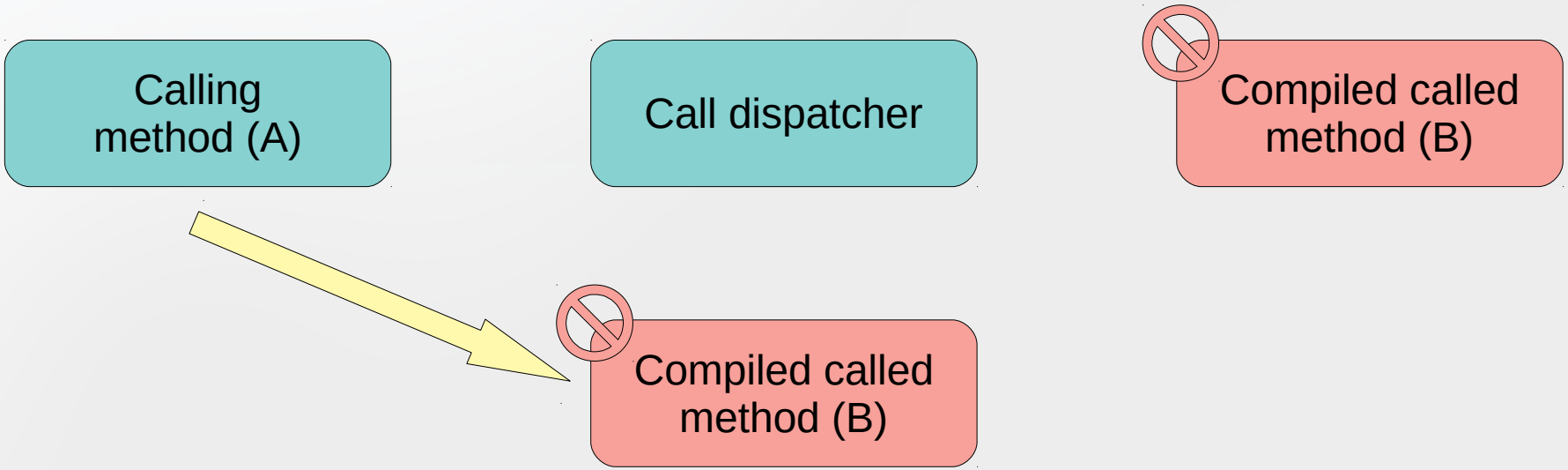
Made not entrant



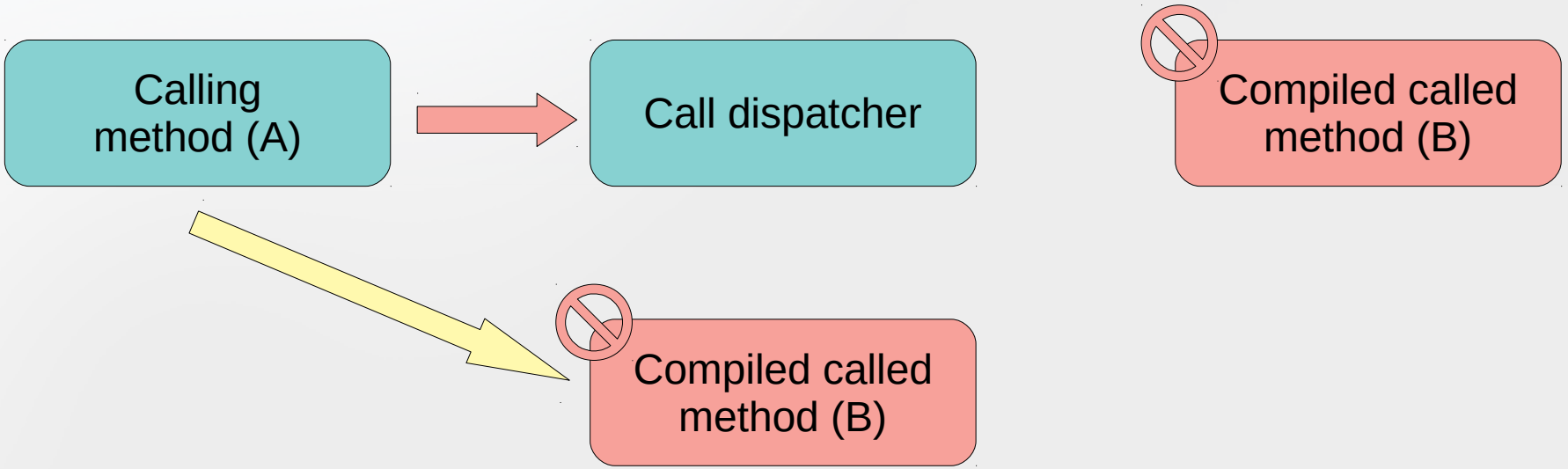
Made not entrant – variant 1



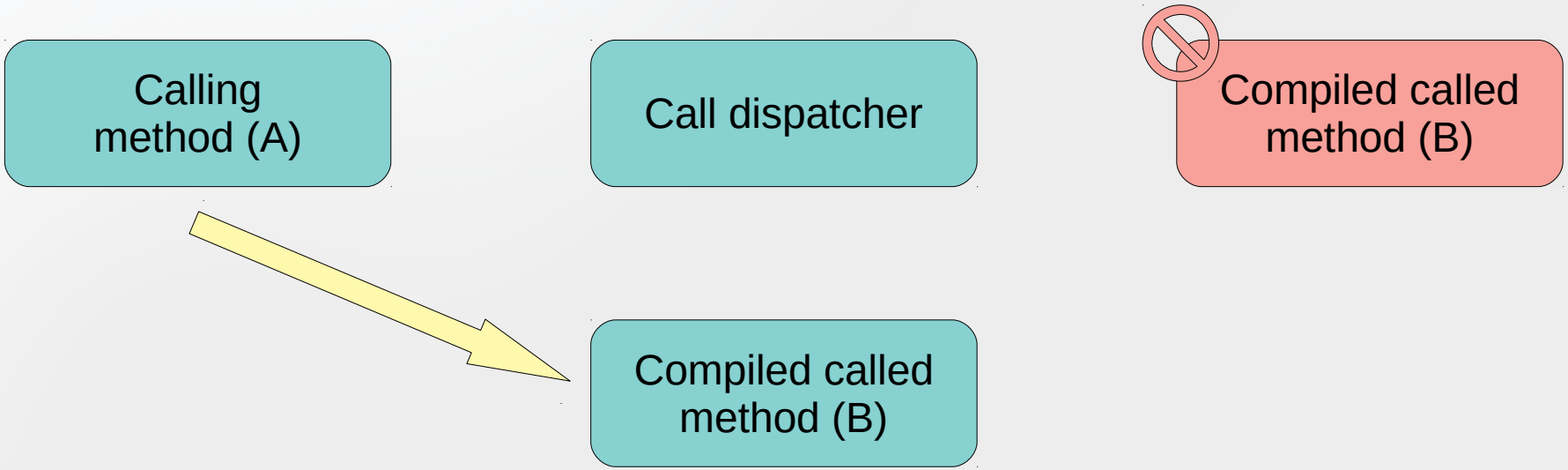
Made not entrant – variant 1



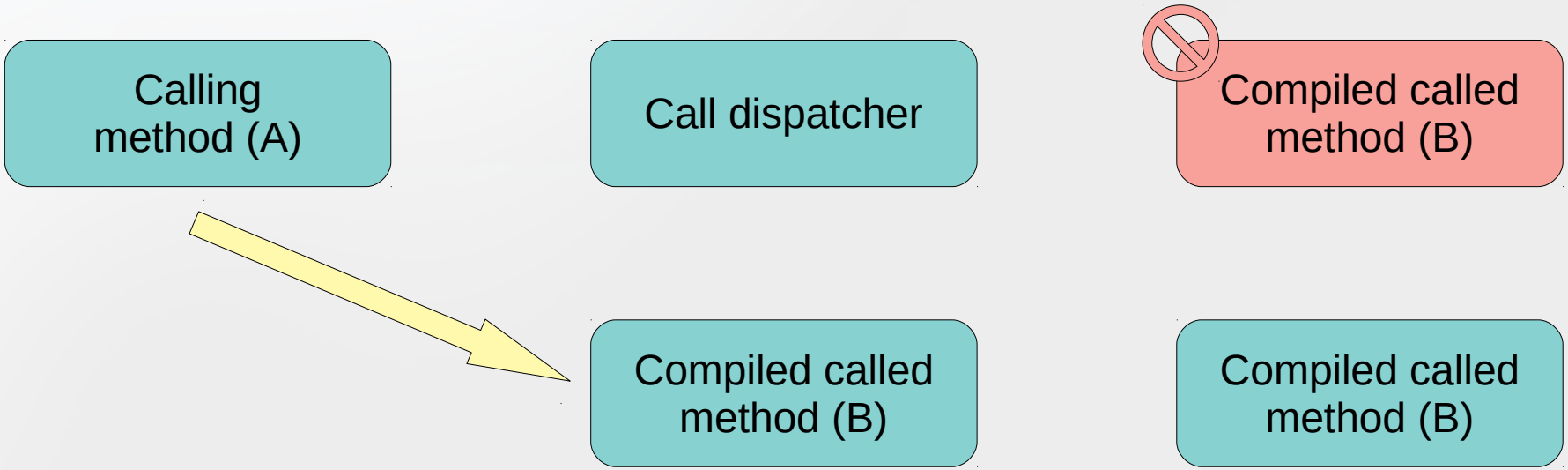
Made not entrant – variant 1



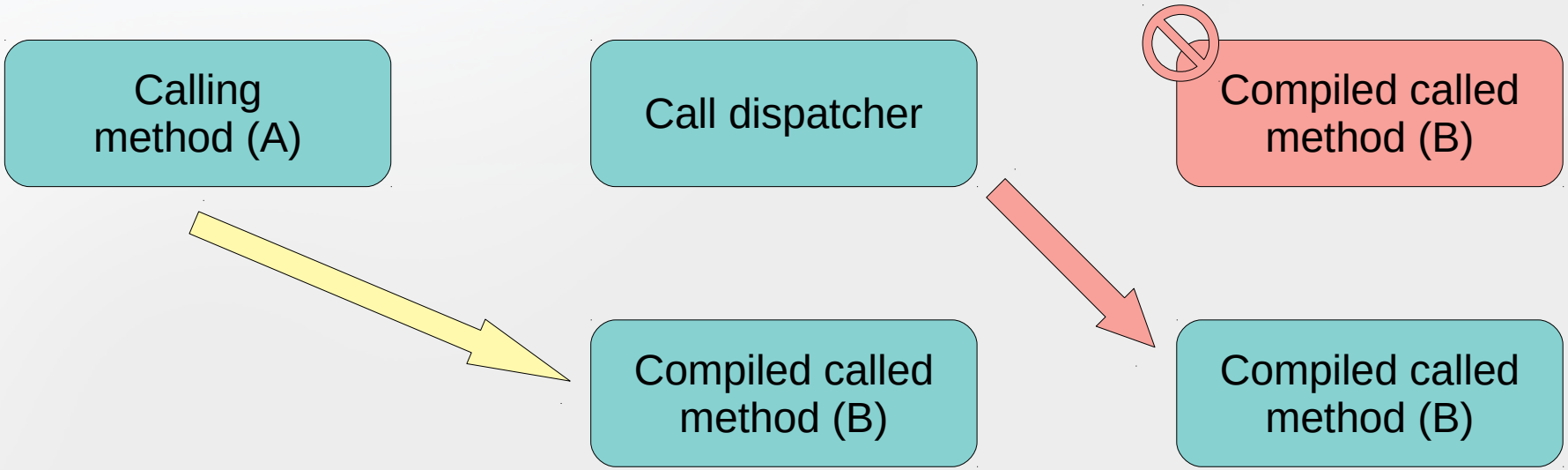
Made not entrant – variant 2



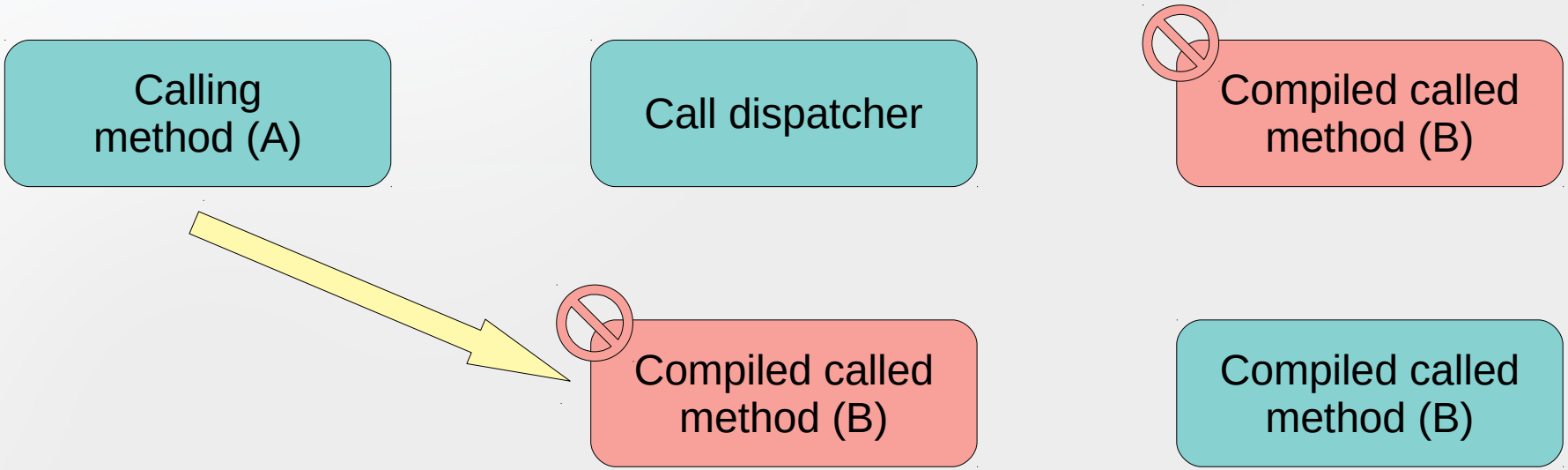
Made not entrant – variant 2



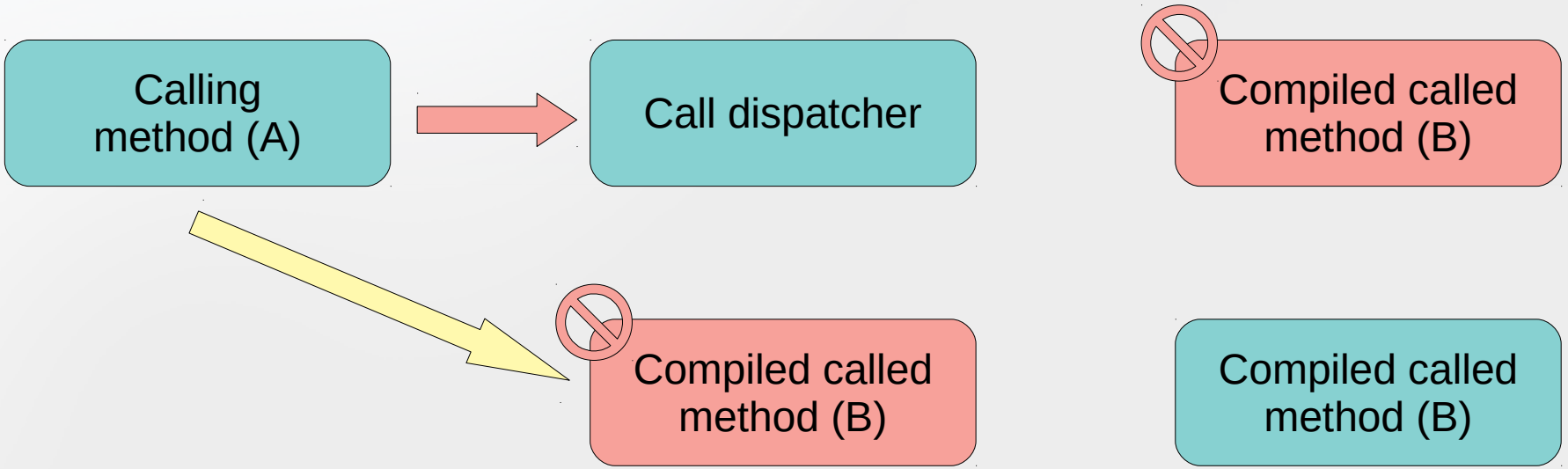
Made not entrant – variant 2



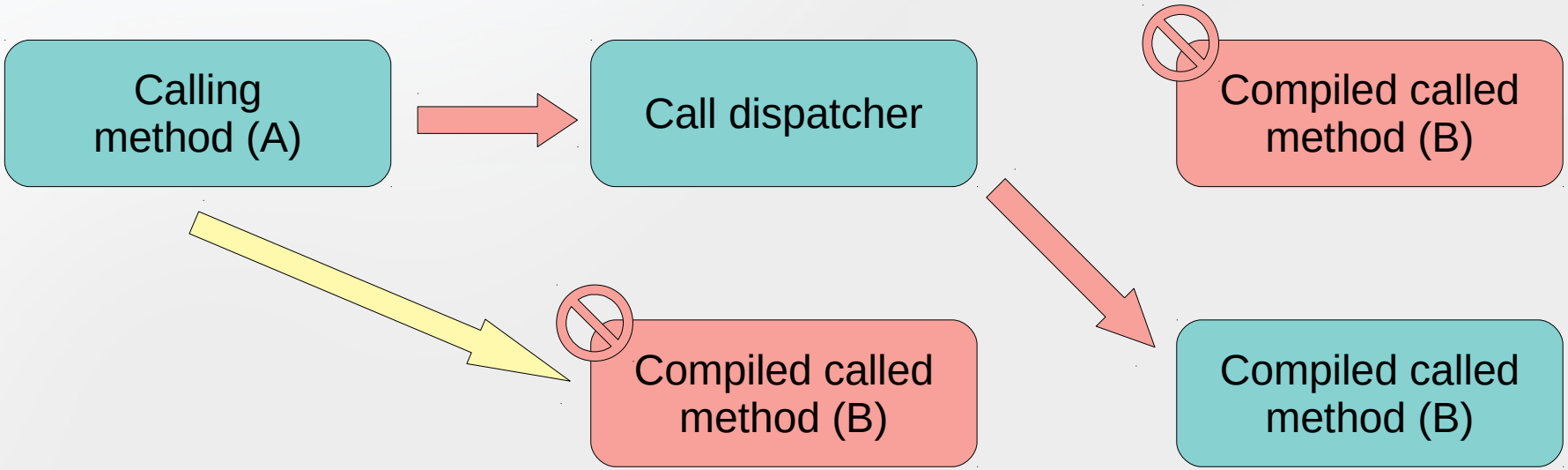
Made not entrant – variant 2



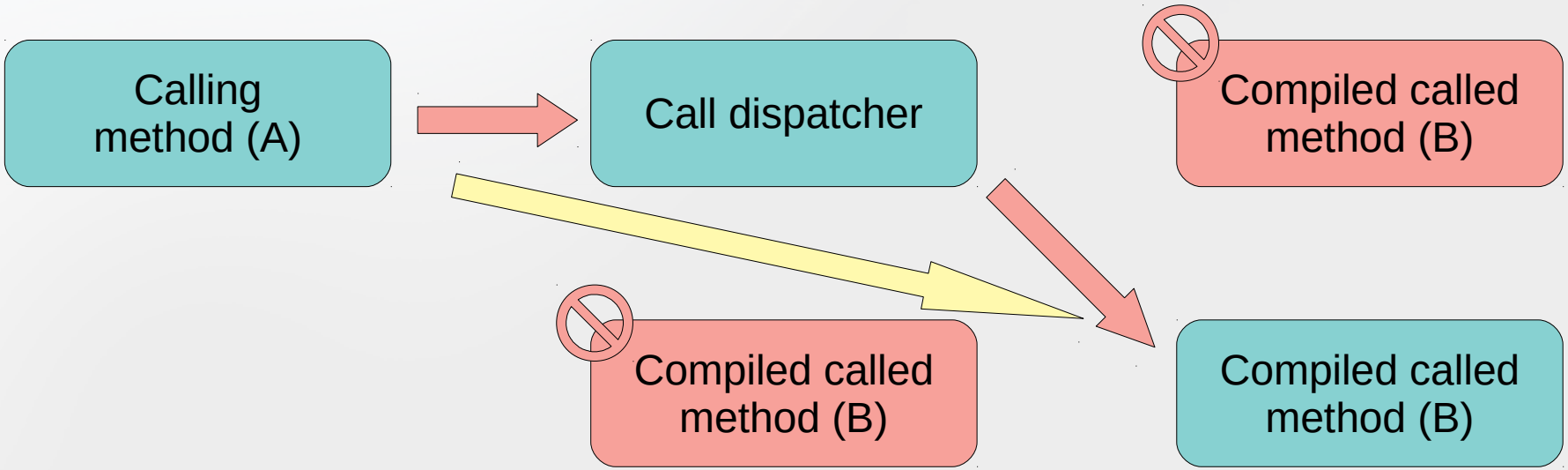
Made not entrant – variant 2



Made not entrant – variant 2



Made not entrant – variant 2



Inlining

- „Matka wszystkich optymalizacji”

Inlining

- „Matka wszystkich optymalizacji”
- Zastąpienie wywołania metody jej „ciałem”

Inlining

- „Matka wszystkich optymalizacji”
- Zastąpienie wywołania metody jej „ciałem”
- Pozwala na spojrzenie z szerszej perspektywy

Inlining

- „Matka wszystkich optymalizacji”
- Zastąpienie wywołania metody jej „ciałem”
- Pozwala na spojrzenie z szerszej perspektywy
- Można śledzić za pomocą
  -XX:+UnlockDiagnosticVMOptions
  -XX:+PrintInlining

Inlining

```
final Object mutex
      = new Object();

void execute() {
    subExecute1(true);
    subExecute2(true);
}
```

```
void subExecute1(boolean mt) {
    if (mt) {
        synchronized (mutex) {
            // logic 1
        }
    } else {
        // logic 1
    }
}
void subExecute2(boolean mt) {
    if (mt) {
        synchronized (mutex) {
            // logic 2
        }
    } else {
        // logic 2
    }
}
```

britenet
quality for lease

Inlining

```
void execute() {
    if (true) {
        synchronized (mutex) {
            // logic 1
        }
    } else {
        // logic 1
    }
    if (true) {
        synchronized (mutex) {
            // logic 2
        }
    } else {
        // logic 2
    }
}
```

Inlining

```
void execute() {
    synchronized (mutex) {
        // logic 1
    }
    synchronized (mutex) {
        // logic 2
    }
}
```

Inlining

```
void execute() {
    synchronized (mutex) {
        // logic 1
        // logic 2
    }
}
```

Intrinsic

- Metody wbudowane

Intrinsic

- Metody wbudowane
- Implementacja podmieniane w:
  - Interpreterze
  - Kompilatorze C1
  - Kompilatorze C2

Intrinsic

- Metody wbudowane
- Implementacja podmieniane w:
  - Interpreterze
  - Kompilatorze C1
  - Kompilatorze C2
- Od JDK 9 oznaczone anotacją `@HotSpotIntrinsicCandidate`

Intrinsic

- Metody wbudowane
- Implementacja podmieniane w:
  - Interpreterze
  - Kompilatorze C1
  - Kompilatorze C2
- Od JDK 9 oznaczone anotacją `@HotSpotIntrinsicCandidate`
- Około 300 metod w JDK

Intrinsic

- Używane do
  - Poprawy wydajności `java.lang.System#arraycopy`

Intrinsic

- Używane do
  - Poprawy wydajności `java.lang.System#arraycopy`
  - Dodania funkcjonalności spoza języka: `sun.misc.Unsafe#allocateInstance`

Intrinsic

- Używane do
  - Poprawy wydajności `java.lang.System#arraycopy`
  - Dodania funkcjonalności spoza języka: `sun.misc.Unsafe#allocateInstance`
  - Bezpośrednie wywołanie instrukcji CPU: `sun.misc.Unsafe#storeFence`

Redundancy removal

- Po co robić 2x to samo?

britenet
quality for lease

Redundancy removal

- Po co robić 2x to samo?

```
static class Value {
    int i;
}

private Value value;

void execute() {
    int j = value.i + 1;
    int k = value.i + 1;
}
```

Implicit checks elimination

- Ochrona przed zrobieniem „głupstwa”

Implicit checks elimination

- Ochrona przed zrobieniem „głupstwa”

```
public static int getSize(Collection collection) {
    return collection.size();
}
```

Implicit checks elimination

- Ochrona przed zrobieniem „głupstwa”

```
public static int getSize(Collection collection) {
    return collection.size();
}
```

To tak na prawdę:

```
public static int getSize(Collection collection) {
    if (collection == null) {
        throw new NullPointerException();
    }
    return collection.size();
}
```

Implicit checks elimination

- Gdyby ich nie było --> crash JVM (np. segmentation fault)

Implicit checks elimination

- Gdyby ich nie było --> crash JVM (np. segmentation fault)
- Optymalizacje dla
  - Dobrych programistów – signal handler

Implicit checks elimination

- Gdyby ich nie było --> crash JVM (np. segmentation fault)
- Optymalizacje dla
  - Dobrych programistów – signal handler
  - Gorszych programistów – check

Implicit checks elimination

- Gdyby ich nie było --> crash JVM (np. segmentation fault)
- Optymalizacje dla
  - Dobrych programistów – signal handler
  - Gorszych programistów – check
  - Bardzo złych programistów – exception cache

Implicit checks elimination

- Gdyby ich nie było --> crash JVM (np. segmentation fault)
- Optymalizacje dla
  - Dobrych programistów – signal handler
  - Gorszych programistów – check
  - Bardzo złych programistów – exception cache
- Optymalizacja spekulatywna

Unused branch removal

- Jeżeli w trakcie profilowania nie wchodzisz do „brancha” to można go usunąć

britenet
quality for lease

Unused branch removal

- Jeżeli w trakcie profilowania nie wchodzisz do „brancha” to można go usunąć
- Optymalizacja spekulatywna

Unused branch removal

```
public void handle(int i) {
    boolean cleanup = false;
    if (i >= 0) {
        // some logic
    } else {
        // some other logic
        cleanup = true;
    }

    if (cleanup) {
        // cleanup logic
    }
}
```

Unused branch removal

```
public void handle(int i) {
    boolean cleanup = false;
    if (i >= 0) {
        // some logic
    } else {
        uncommon_trap
    }

    if (cleanup) {
        uncommon_trap
    }
}
```

Unused branch removal

```
public void handle(int i) {
    boolean cleanup = false;
    if (i >= 0) {
        // some logic
    } else {
        uncommon_trap
    }
}
```

Devirtualization

- Pozbycie się wirtualnych calli

Devirtualization

- Pozbycie się wirtualnych calli
- Inna optymalizacja dla
  - Klas – Class Hierarchy Analysis + Type Profile
  - Interfejsów – Type Profile

Devirtualization

- Pozbycie się wirtualnych calli
- Inna optymalizacja dla
  - Klas – Class Hierarchy Analysis + Type Profile
  - Interfejsów – Type Profile
- Optymalizacja spekulatywna

Devirtualization – klasy

```
abstract class AbstractMathFunc {
    abstract double apply(double i);
}
class CosFunc extends AbstractMathFunc {
    double apply(double i) {
        return Math.cos(i);
    }
}
class SinFunc extends AbstractMathFunc {
    double apply(double i) {
        return Math.sin(i);
    }
}
class SqrtFunc extends AbstractMathFunc {
    double apply(double i) {
        return Math.sqrt(i);
    }
}
```

Devirtualization – klasy

```
static double do1(AbstractMathFunc func, double i) {
    return func.apply(i);
}
```

Devirtualization – klasy

```
static double do1(AbstractMathFunc func, double i) {
    return func.apply(i);
}
```

- Co jeżeli jest tylko SinFunc?

Devirtualization – klasy – 1

```
static double do1(AbstractMathFunc func, double i) {
    return Math.sin(i);
}
```

Devirtualization – klasy – 1

```
static double do1(AbstractMathFunc func, double i) {
    return Math.sin(i);
}
```

- Co jeżeli jest pojawi się CosFunc?

Devirtualization – klasy – 2

```
static double do1(AbstractMathFunc func, double i) {
    if (func.getClass().equals(SinFunc.class)) {
        return Math.sin(i);
    } else {
        uncommon_trap
    }
}
```

Devirtualization – klasy – 2

```
static double do1(AbstractMathFunc func, double i) {
    if (func.getClass().equals(SinFunc.class)) {
        return Math.sin(i);
    } else {
        uncommon_trap
    }
}
```

- Co jeżeli jest CosFunc zacznie być używane?

Devirtualization – klasy – 3

```
static double do1(AbstractMathFunc func, double i) {
    if (func.getClass().equals(SinFunc.class)) {
        return Math.sin(i);
    } else if (func.getClass().equals(CosFunc.class)) {
        return Math.cos(i);
    } else {
        uncommon_trap
    }
}
```

Devirtualization – klasy – 3

```
static double do1(AbstractMathFunc func, double i) {
    if (func.getClass().equals(SinFunc.class)) {
        return Math.sin(i);
    } else if (func.getClass().equals(CosFunc.class)) {
        return Math.cos(i);
    } else {
        uncommon_trap
    }
}
```

- Co jeżeli jest pojawi się SqrtFunc?

Devirtualization – klasy – 3

```
static double do1(AbstractMathFunc func, double i) {
    if (func.getClass().equals(SinFunc.class)) {
        return Math.sin(i);
    } else if (func.getClass().equals(CosFunc.class)) {
        return Math.cos(i);
    } else {
        uncommon_trap
    }
}
```

- Co jeżeli jest pojawi się SqrtFunc?
- Co jeżeli jest SqrtFunc zacznie być używane?

Devirtualization – klasy – 4

```
static double do1(AbstractMathFunc func, double i) {
    return func.apply(i);
}
```

Inne optymalizacje

- Dead code elimination
- Locks:
  - Biased locking
  - Lock coarsening
  - Lock elision
  - Adaptive locking
- Loops:
  - Loop unrolling
  - Loop peeling
- ...

Konsekwencje

- Możliwe fazy stop the world

Konsekwencje

- Możliwe fazy stop the world
- Exception cache

Konsekwencje

- Możliwe fazy stop the world
- Exception cache
- Kod ma znaczenie

Co lubi JIT?

- Krótkie metody

Co lubi JIT?

- Krótkie metody
- Zmienne lokalne

Co lubi JIT?

- Krótkie metody
- Zmienne lokalne
- Niemutowalność

Co lubi JIT?

- Krótkie metody
- Zmienne lokalne
- Niemutowalność
- Ładny, przejrzysty kod

Co lubi JIT?

- Krótkie metody
- Zmienne lokalne
- Niemutowalność
- Ładny, przejrzysty kod
- Typy prymitywne

Co lubi JIT?

- Krótkie metody
- Zmienne lokalne
- Niemutowalność
- Ładny, przejrzysty kod
- Typy prymitywne
- Abstrakcje z jedną/dwiema implementacją

Co lubi JIT?

- Krótkie metody
- Zmienne lokalne
- Niemutowalność
- Ładny, przejrzysty kod
- Typy prymitywne
- Abstrakcje z jedną/dwiema implementacją
- Nieprzeciążone metody

Jaki kod lubi JIT?

```
void execute() {
    // ... some logic
    int result = send();
    if (result < 0) {
        // handle error
        // ... very long logic
    }
}
```

Jaki kod lubi JIT?

```
void execute() {
    // ... some logic
    int result = send();
    if (result < 0) {
        // handle error
        handleError();
    }
}
```

Konsekwencje

- Możliwe fazy stop the world
- Exception cache
- Kod ma znaczenie
- Składnia vs wydajność,
  struktury danych vs wydajność

Wydajność

britenet
quality for lease

Wydajność

Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We should forget about small efficiencies, say about 97% of the time:

premature optimization is the root of all evil. *Yet we should not pass up our opportunities in that critical 3%.*

Donald Knuth

britenet
quality for lease

Wydajność

britenet
quality for lease

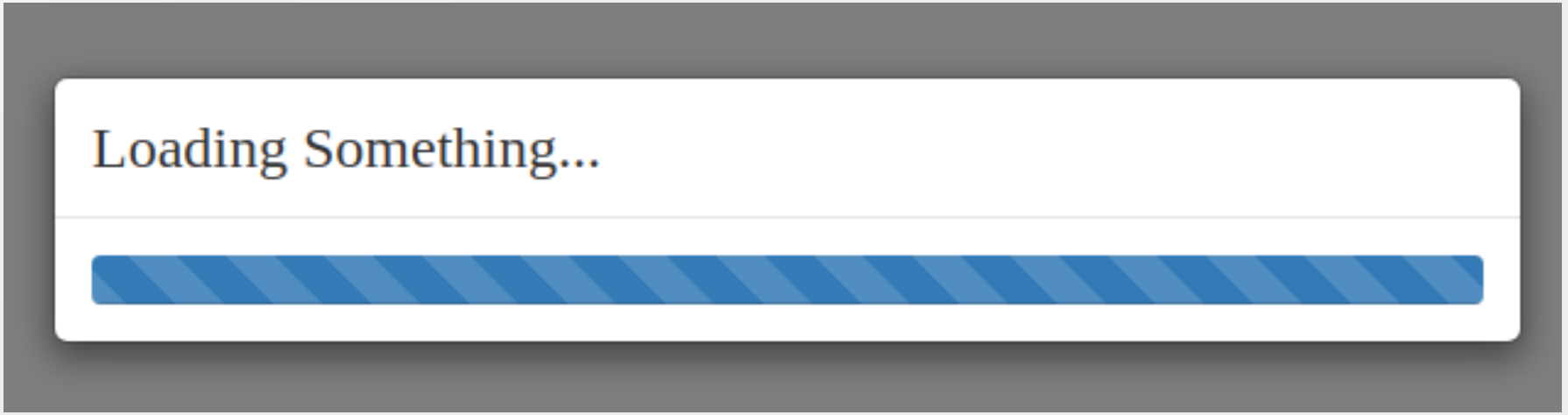
Wydajność

Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%.

Donald Knuth

Wydajność

Wydajność



britenet
quality for lease

Konsekwencje

- Możliwe fazy stop the world
- Exception cache
- Kod ma znaczenie
- Składnia vs wydajność,
  struktury danych vs wydajność
- Restart JVM --> zaczynamy od nowa

Konsekwencje

- Możliwe fazy stop the world
- Exception cache
- Kod ma znaczenie
- Składnia vs wydajność,
  struktury danych vs wydajność
- Restart JVM --> zaczynamy od nowa
- Concurrency?

Concurrency

- „Obsługa wielu wątków w Javie to suka ...”

Concurrency

- „Obsługa wielu wątków w Javie to suka ...”
- „… wszystkiemu winny jest JIT ...”

Concurrency

- „Obsługa wielu wątków w Javie to suka ...”
- „… wszystkiemu winny jest JIT ...”
- „… i nowoczesny hardware ...”

Concurrency

- „Obsługa wielu wątków w J… to sukces …”
- „… wszystkiemu winny jest J…”
- „… i nowoczesny hardware …”

britenet
quality for lease

Concurrency

An implementation is free to produce any code it likes, as long as all resulting executions of a program produce a result that can be predicted by the memory model. This provides a great deal of freedom for the implementor to perform a myriad of code transformations, ***including the reordering of actions*** *and removal of unnecessary synchronization.*

Java Language Specification (17.4)

Concurrency

```
private int i;
private int j;
private int k;

void execute() {
    i = 1;
    j = 2;
    k = 3;
    if (i > 0) {
        i = k;
    }
}
```

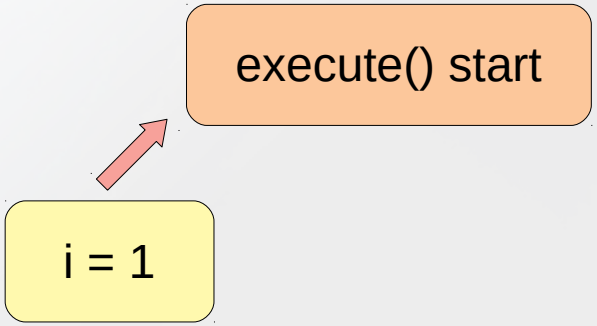
Concurrency

```
private int i;
private int j;
private int k;

void execute() {
    i = 1;
    j = 2;
    k = 3;
    if (i > 0) {
        i = k;
    }
}
```

execute() start

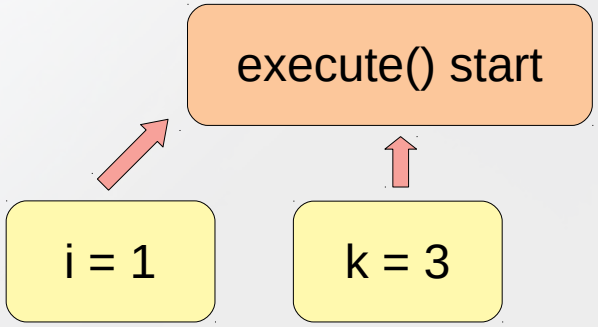
Concurrency

```
private int i;
private int j;
private int k;

void execute() {
    i = 1;
    j = 2;
    k = 3;
    if (i > 0) {
        i = k;
    }
}
```



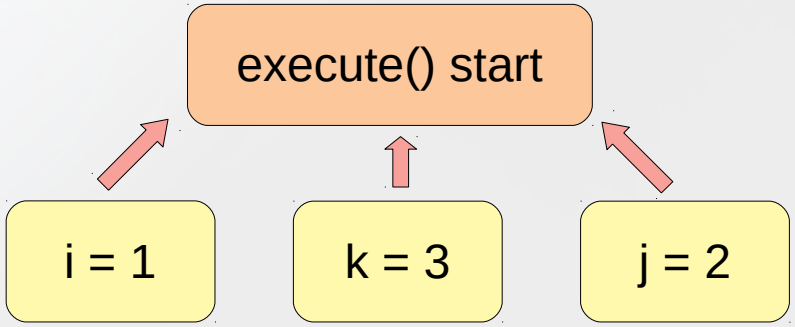
Concurrency

```
private int i;
private int j;
private int k;

void execute() {
    i = 1;
    j = 2;
    k = 3;
    if (i > 0) {
        i = k;
    }
}
```



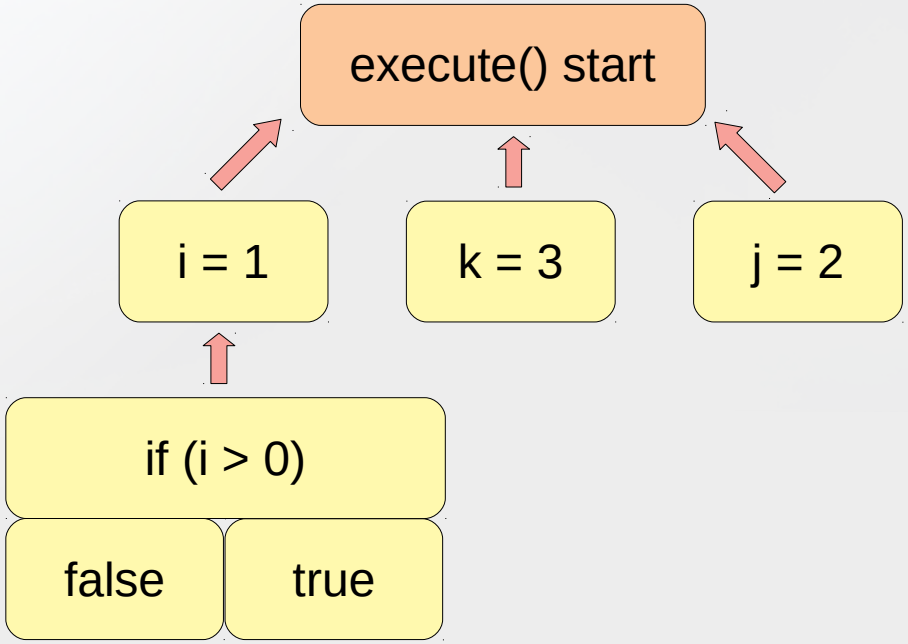
Concurrency

```
private int i;
private int j;
private int k;

void execute() {
    i = 1;
    j = 2;
    k = 3;
    if (i > 0) {
        i = k;
    }
}
```



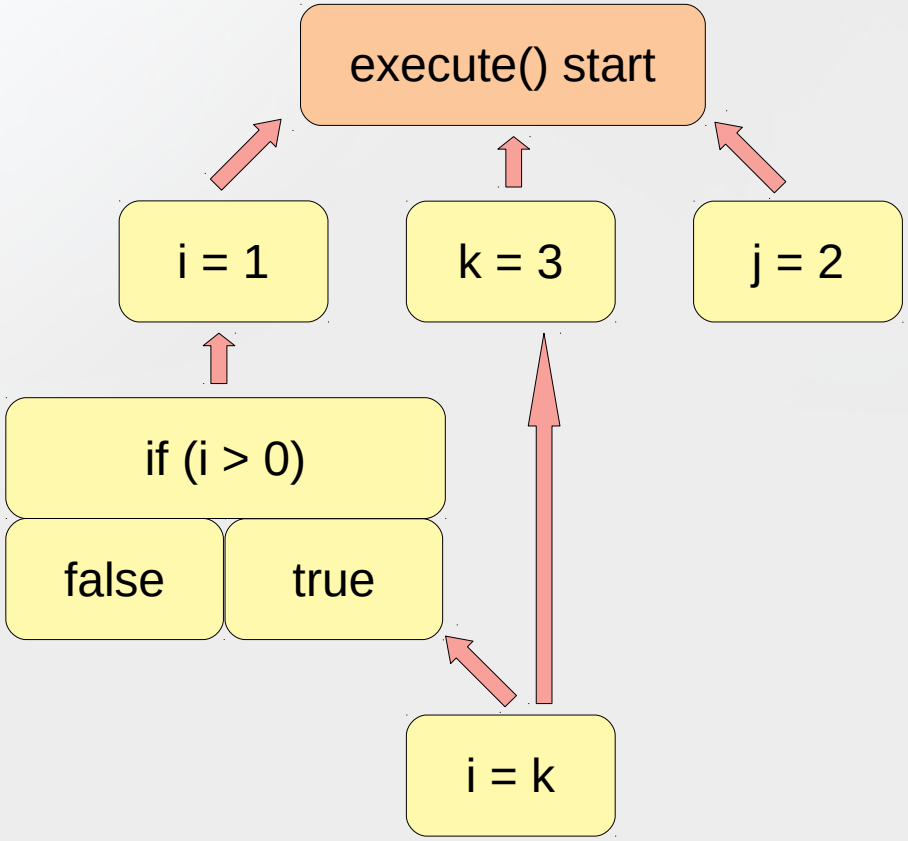
Concurrency

```
private int i;
private int j;
private int k;

void execute() {
    i = 1;
    j = 2;
    k = 3;
    if (i > 0) {
        i = k;
    }
}
```



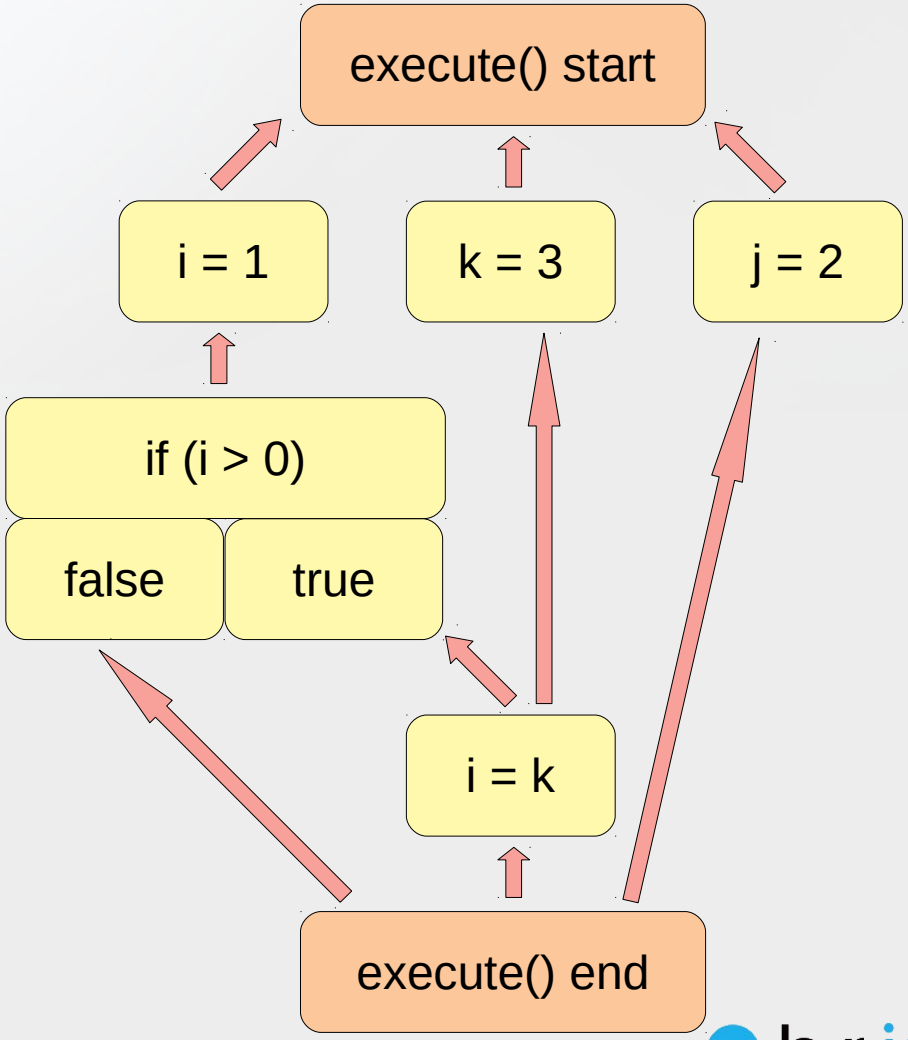
Concurrency

```
private int i;
private int j;
private int k;

void execute() {
    i = 1;
    j = 2;
    k = 3;
    if (i > 0) {
        i = k;
    }
}
```



Concurrency

```
private int i;
private int j;
private int k;

void execute() {
    i = 1;
    j = 2;
    k = 3;
    if (i > 0) {
        i = k;
    }
}
```



Double-checked locking

```
private Helper helper;

public Helper getHelper() {
    if (helper == null) {
        synchronized (this) {
            if (helper == null) {
                helper = new Helper();
            }
        }
    }
    return helper;
}
```

Double-checked locking

```
private Helper helper;

public Helper getHelper() {
    if (helper == null) {
        synchronized (this) {
            if (helper == null) {
                local = calloc(sizeof(Helper.class));
                local.<init>();
                helper = local;
            }
        }
    }
    return helper;
}
```

Double-checked locking

```
synchronized (this) {
 if (helper == null) {
  local = c(s(Helper.class));
  local.<init>();
  helper = local;
 }
}
```

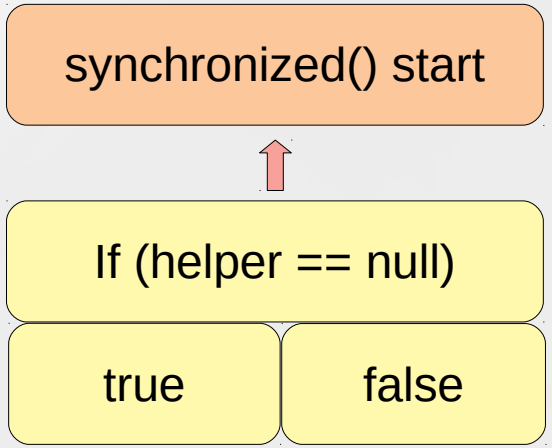
Double-checked locking

```
synchronized (this) {
 if (helper == null) {
  local = c(s(Helper.class));
  local.<init>();
  helper = local;
 }
}
```

synchronized() start

Double-checked locking

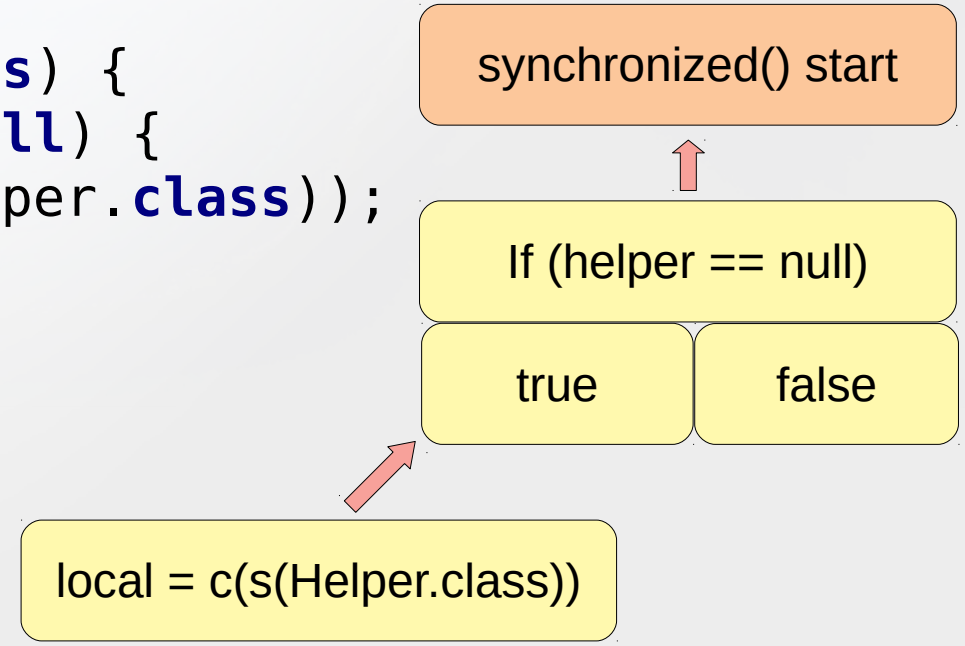
```
synchronized (this) {
 if (helper == null) {
  local = c(s(Helper.class));
  local.<init>();
  helper = local;
 }
}
```



britenet
quality for lease

Double-checked locking

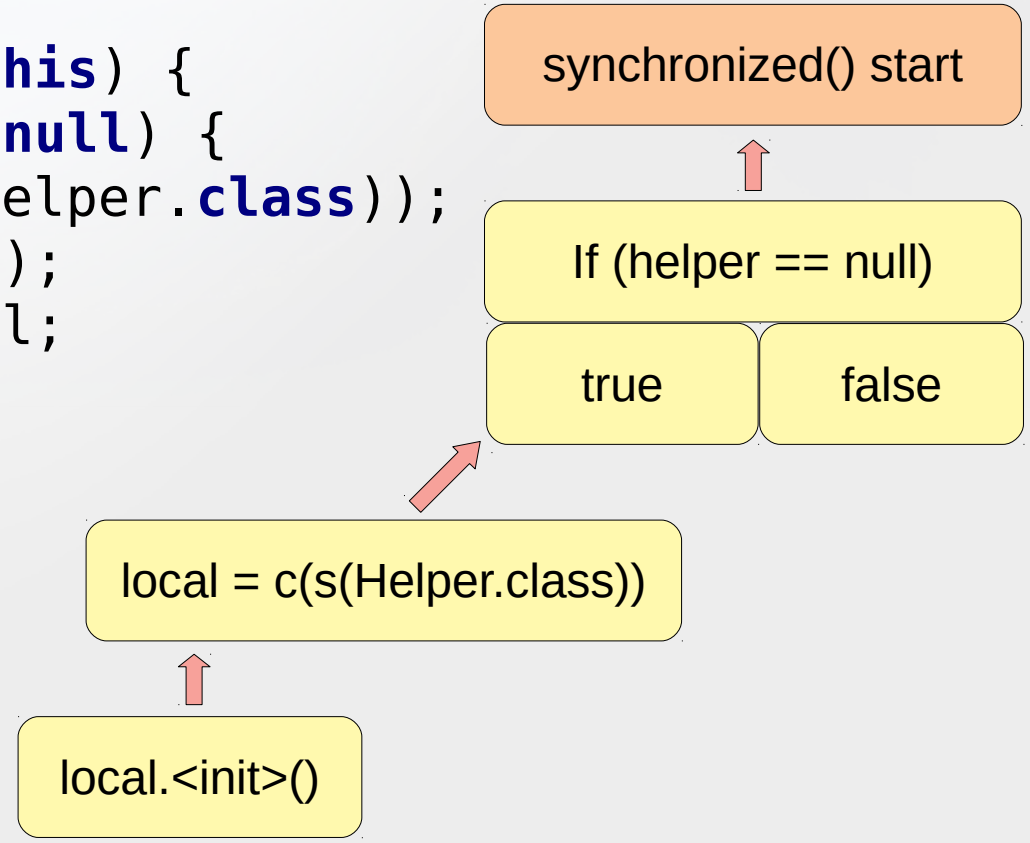
```
synchronized (this) {
 if (helper == null) {
  local = c(s(Helper.class));
  local.<init>();
  helper = local;
 }
}
```



britenet
quality for lease

Double-checked locking

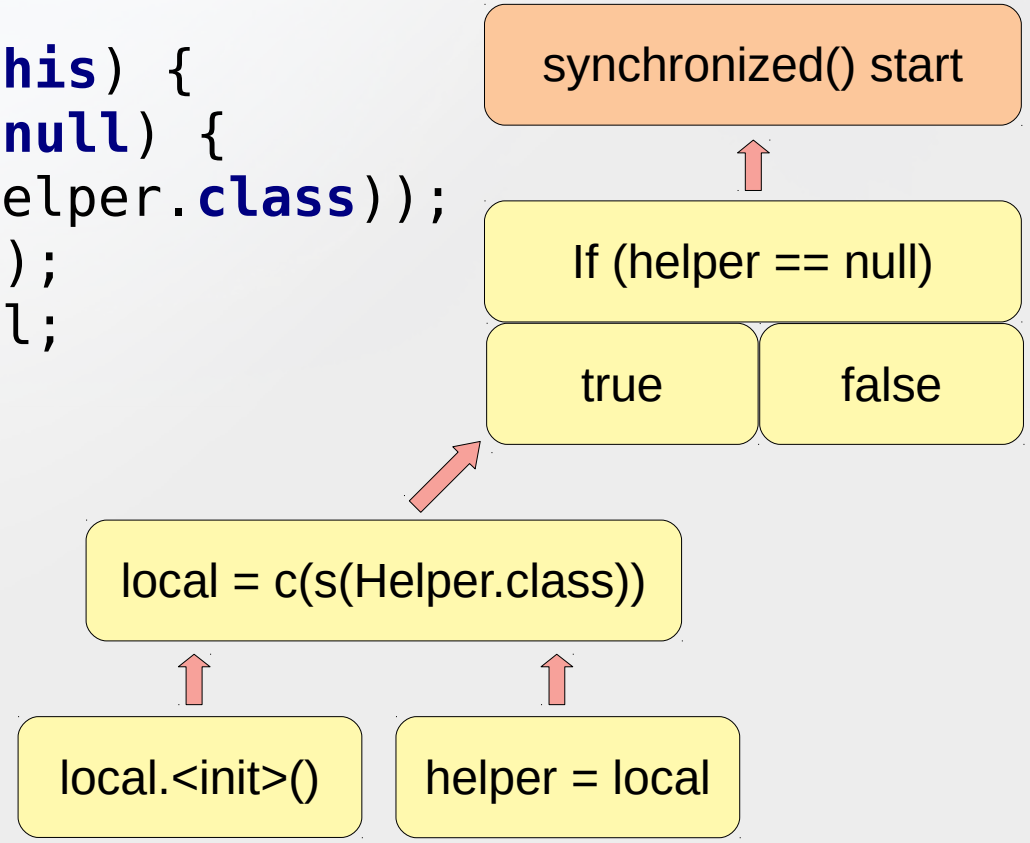
```
synchronized (this) {
 if (helper == null) {
  local = c(s(Helper.class));
  local.<init>();
  helper = local;
 }
}
```



britenet
quality for lease

Double-checked locking

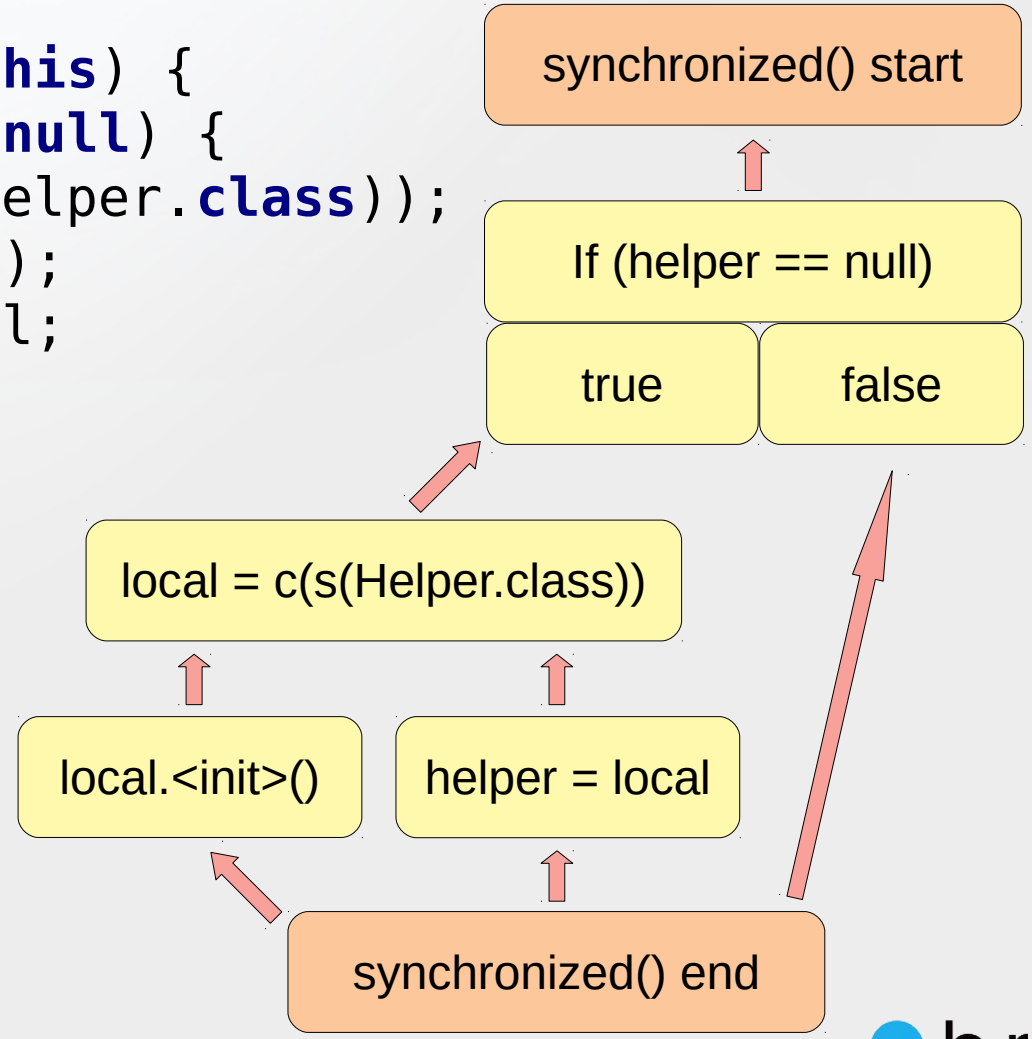
```
synchronized (this) {
 if (helper == null) {
  local = c(s(Helper.class));
  local.<init>();
  helper = local;
 }
}
```



britenet
quality for lease

Double-checked locking

```
synchronized (this) {
 if (helper == null) {
  local = c(s(Helper.class));
  local.<init>();
  helper = local;
 }
}
```



britenet
quality for lease

Double-checked locking

```
private Helper helper;

public Helper getHelper() {
    if (helper == null) {
        synchronized (this) {
            if (helper == null) {
                helper = new Helper();
            }
        }
    }
    return helper;
}
```

Concurrency

*The behavior of threads, particularly when **not correctly synchronized**, can be confusing and counterintuitive.*

*The semantics of the Java programming language allow compilers and microprocessors to perform optimizations that can interact with **incorrectly synchronized** code in ways that can produce behaviors that seem paradoxical.*

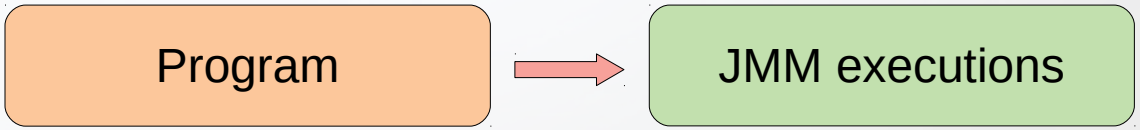
*To some programmers, this behavior may seem "broken". However, it should be noted thatthis code is **improperly synchronize**.*

Java Language Specification (17 & 17.4)

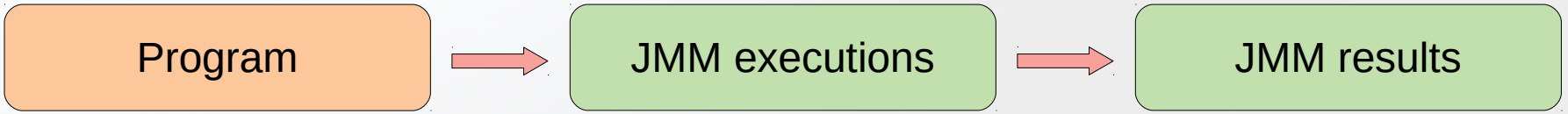
Spec vs impl

Program

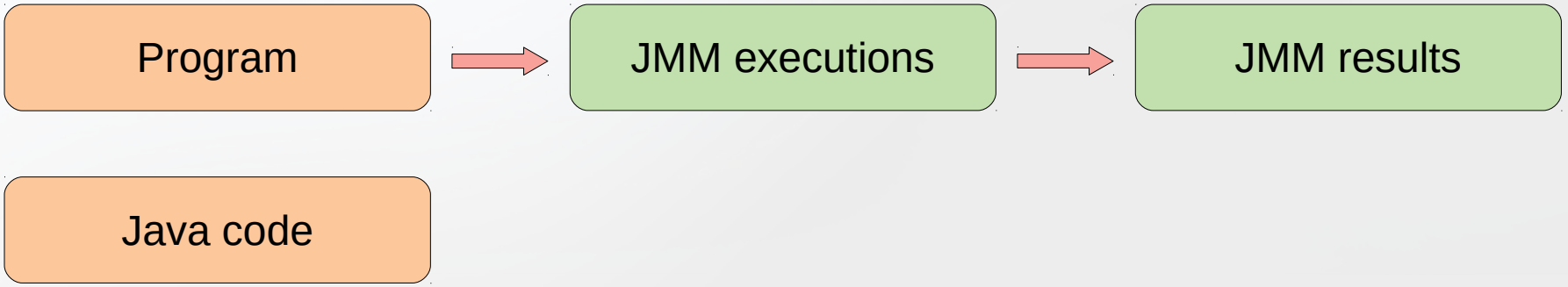
Spec vs impl



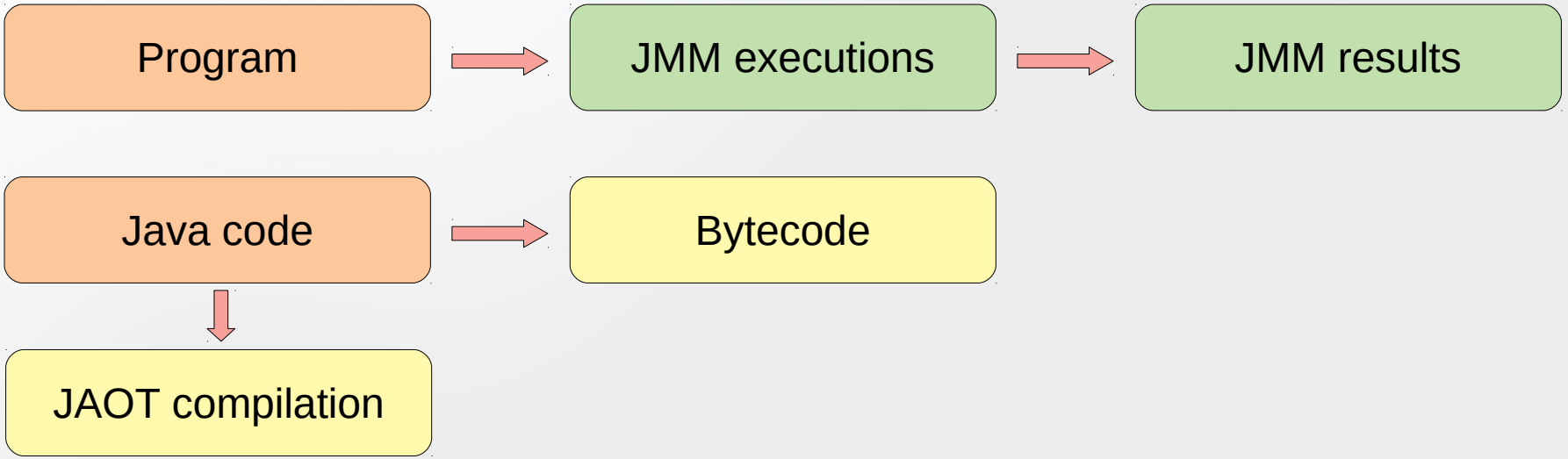
Spec vs impl



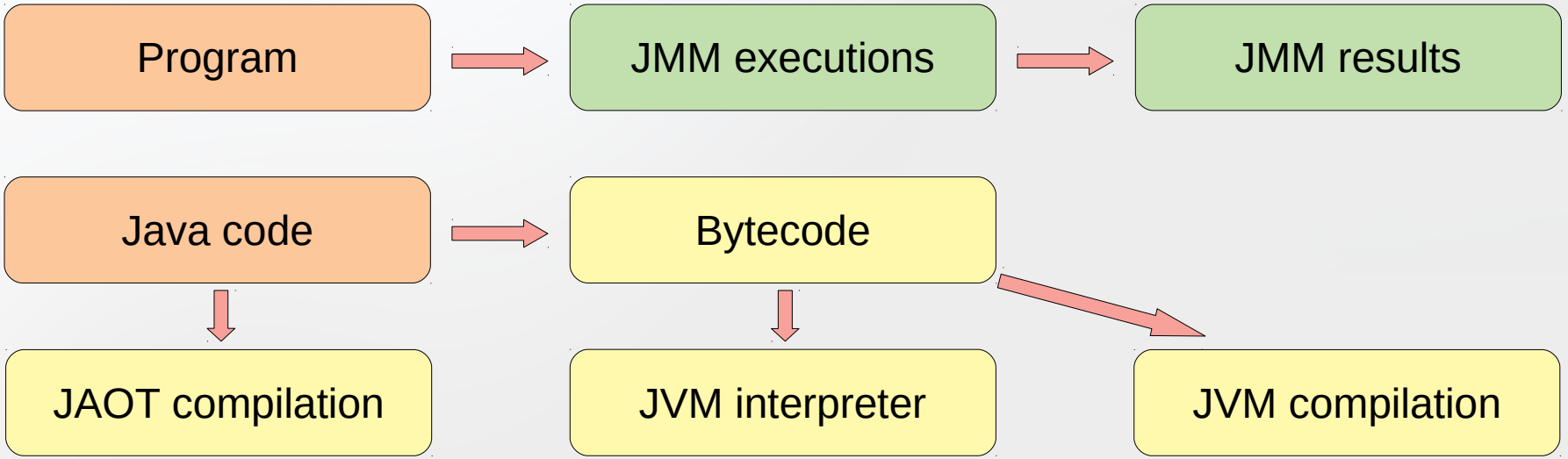
Spec vs impl



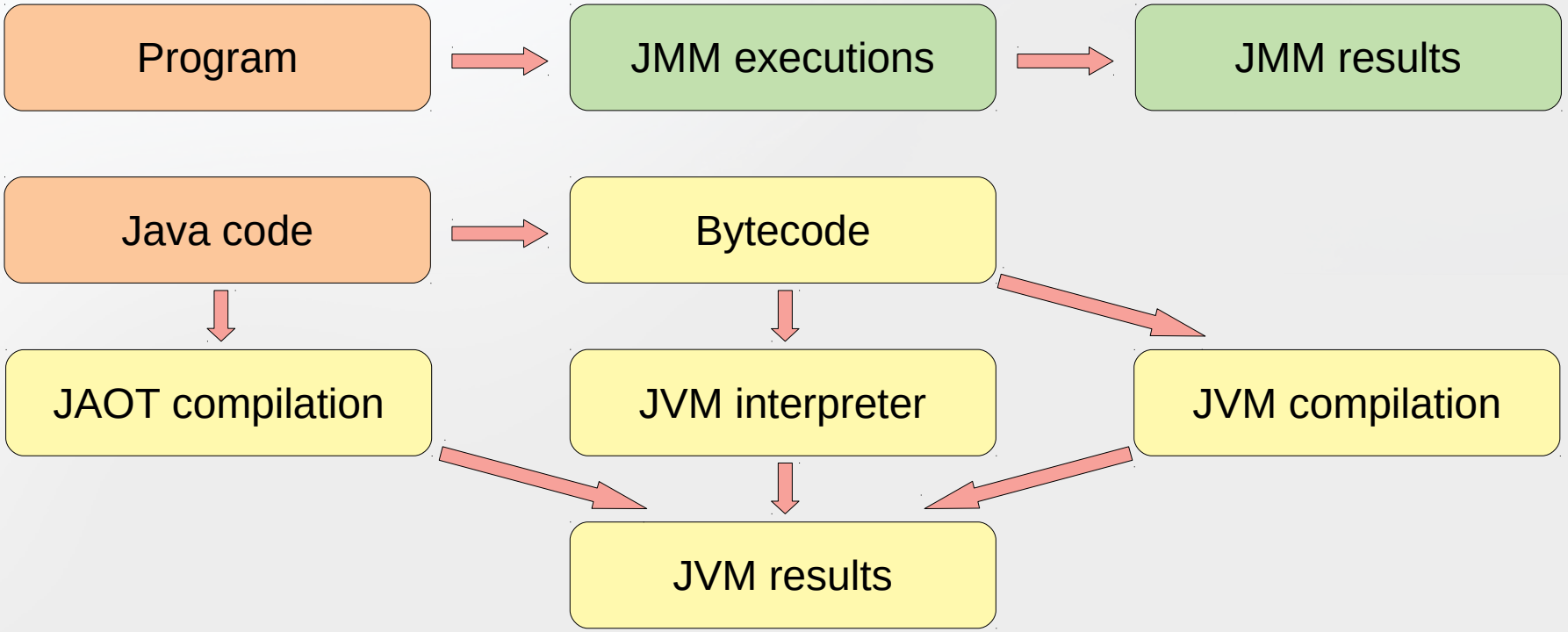
Spec vs impl



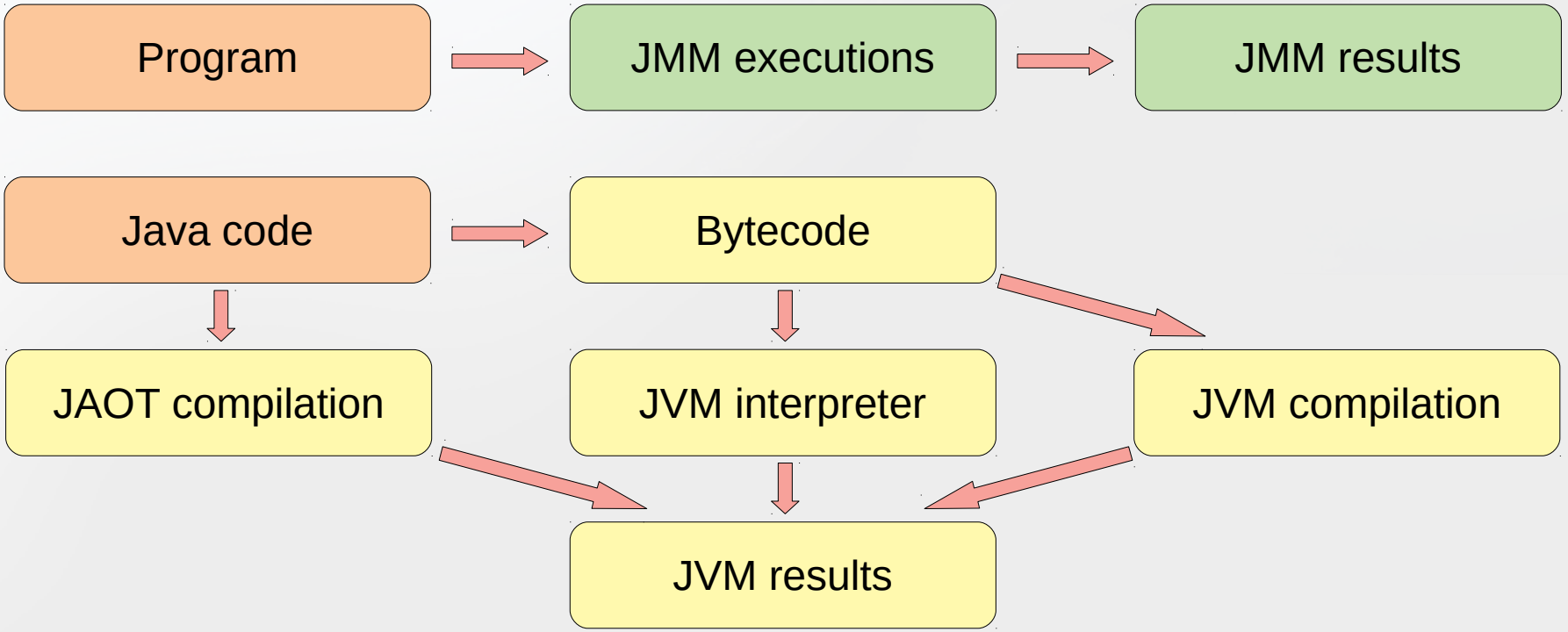
Spec vs impl



Spec vs impl



Spec vs impl



- JVM results to podzbiór JMM results

Concurrency

An implementation is free to produce any code it likes, ***as long as all resulting executions of a program produce a result that can be predicted by the memory model****. This provides a great deal of freedom for the implementor to perform a myriad of code transformations, including the reordering of actions and removal of unnecessary synchronization.*

Java Language Specification (17.4)

JMM w pigułce – 1 / 2

- Obiekt immutable: (Safe construction)

JMM w pigułce – 1 / 2

- Obiekt immutable: (Safe construction)
  - Wszystkie pola final

JMM w pigułce – 1 / 2

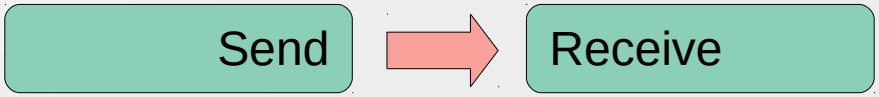
- Obiekt immutable: (Safe construction)
  - Wszystkie pola final
  - Nie udostępnia samego siebie w konstruktorze

JMM w pigułce – 1 / 2

- Obiekt immutable: (Safe construction)
  - Wszystkie pola final
  - Nie udostępnia samego siebie w konstruktorze
- Pary synchronizacji:

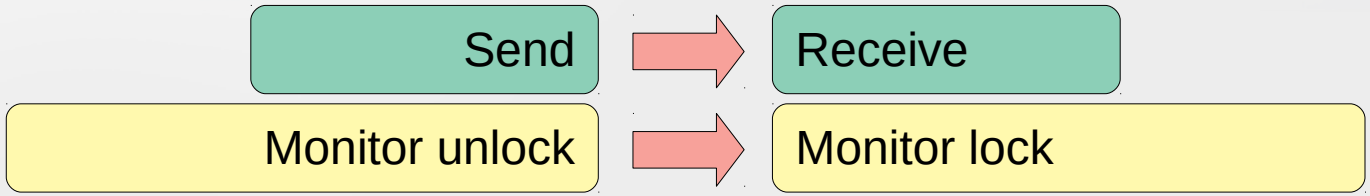
JMM w pigułce – 1 / 2

- Obiekt immutable: (Safe construction)
  - Wszystkie pola final
  - Nie udostępnia samego siebie w konstruktorze
- Pary synchronizacji:



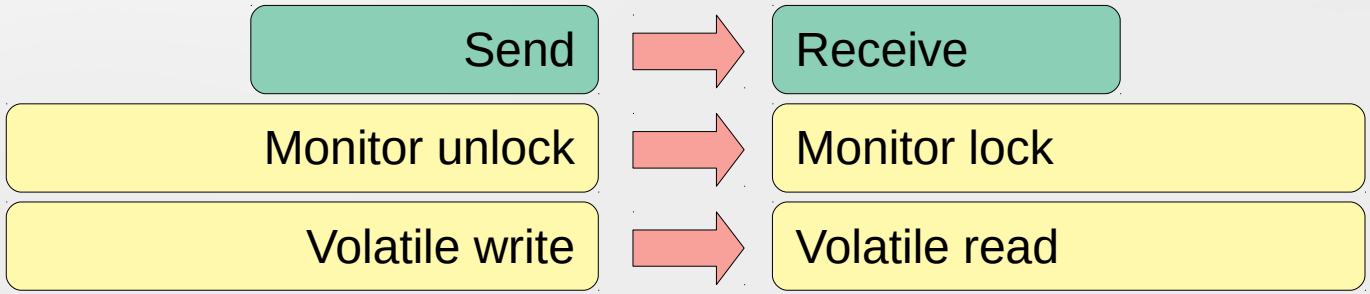
JMM w pigułce – 1 / 2

- Obiekt immutable: (Safe construction)
  - Wszystkie pola final
  - Nie udostępnia samego siebie w konstruktorze
- Pary synchronizacji:



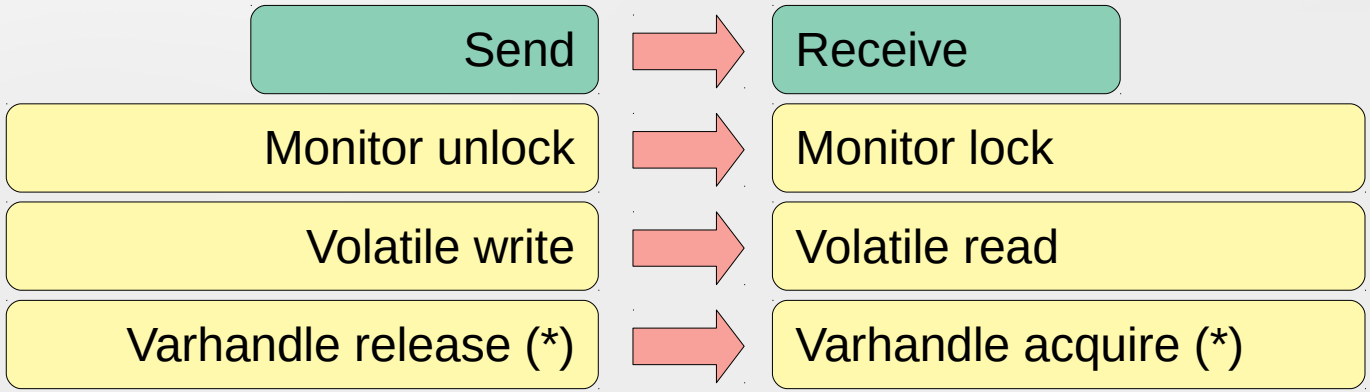
JMM w pigułce – 1 / 2

- Obiekt immutable: (Safe construction)
  - Wszystkie pola final
  - Nie udostępnia samego siebie w konstruktorze
- Pary synchronizacji:



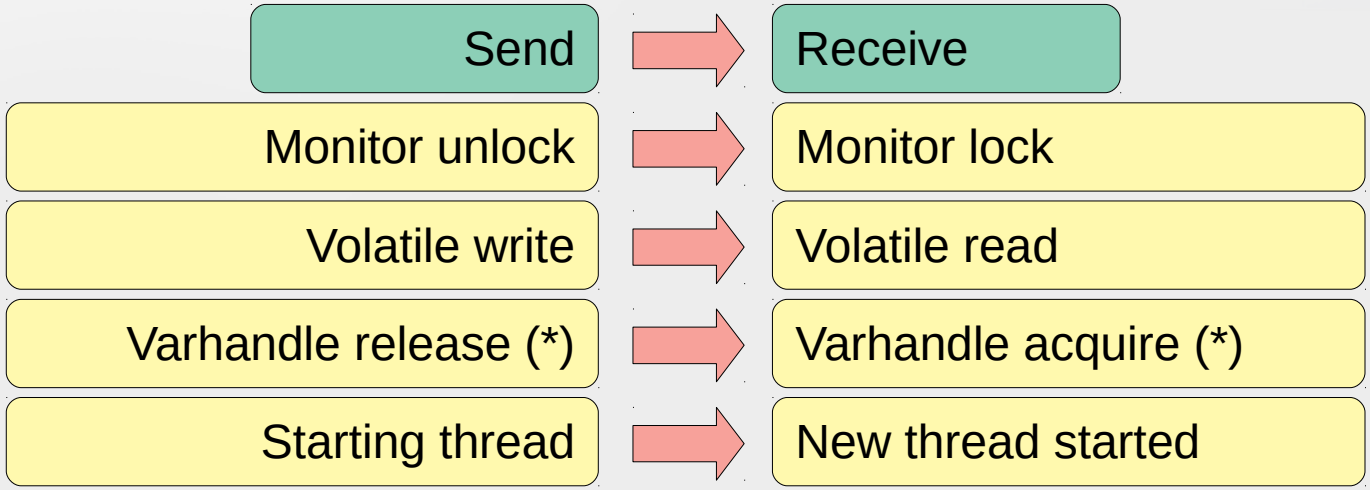
JMM w pigułce – 1 / 2

- Obiekt immutable: (Safe construction)
  - Wszystkie pola final
  - Nie udostępnia samego siebie w konstruktorze
- Pary synchronizacji:



JMM w pigułce – 1 / 2

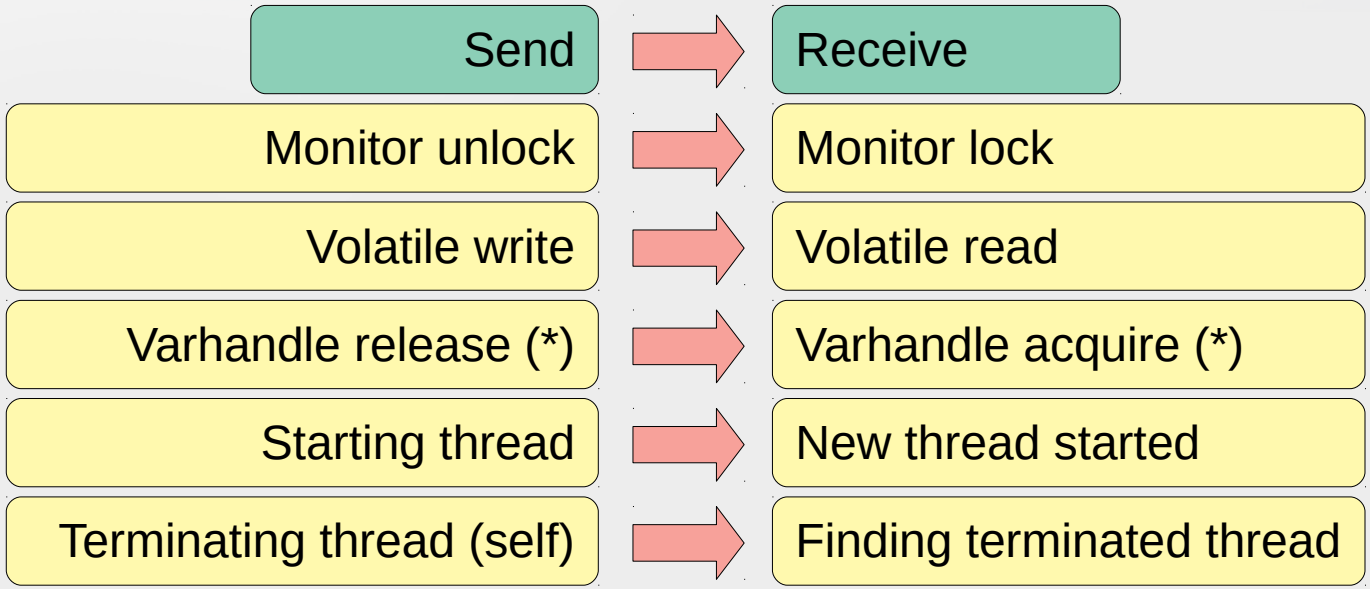
- Obiekt immutable: (Safe construction)
  - Wszystkie pola final
  - Nie udostępnia samego siebie w konstruktorze
- Pary synchronizacji:



britenet
quality for lease

JMM w pigułce – 1 / 2

- Obiekt immutable: (Safe construction)
  - Wszystkie pola final
  - Nie udostępnia samego siebie w konstruktorze
- Pary synchronizacji:



JMM w pigułce – 2 / 2

Thread 1

Write a

...

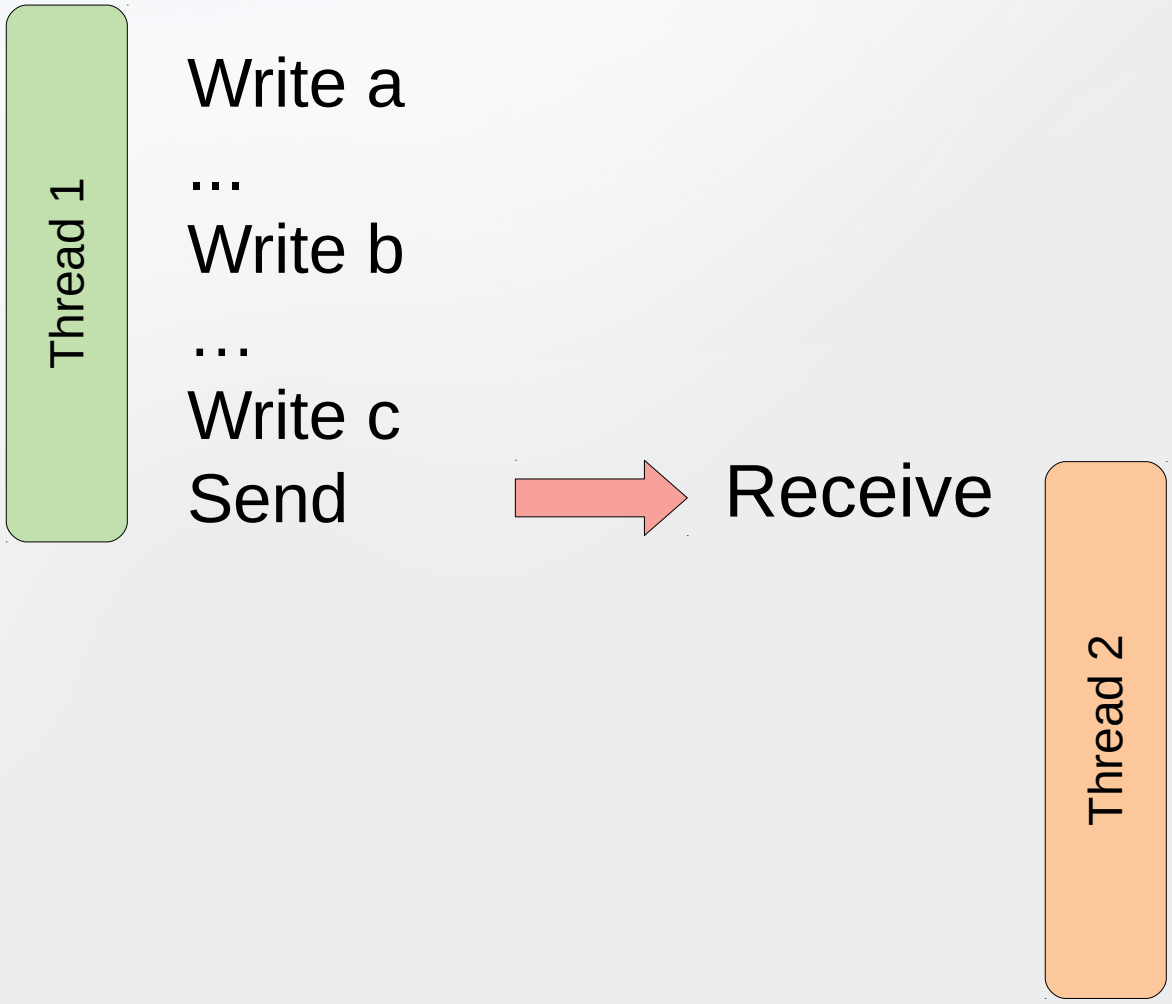
Write b

...

Write c

Thread 2

JMM w pigułce – 2 / 2



JMM w pigułce – 2 / 2

Thread 1

Write a
...
Write b
...
Write c
Send

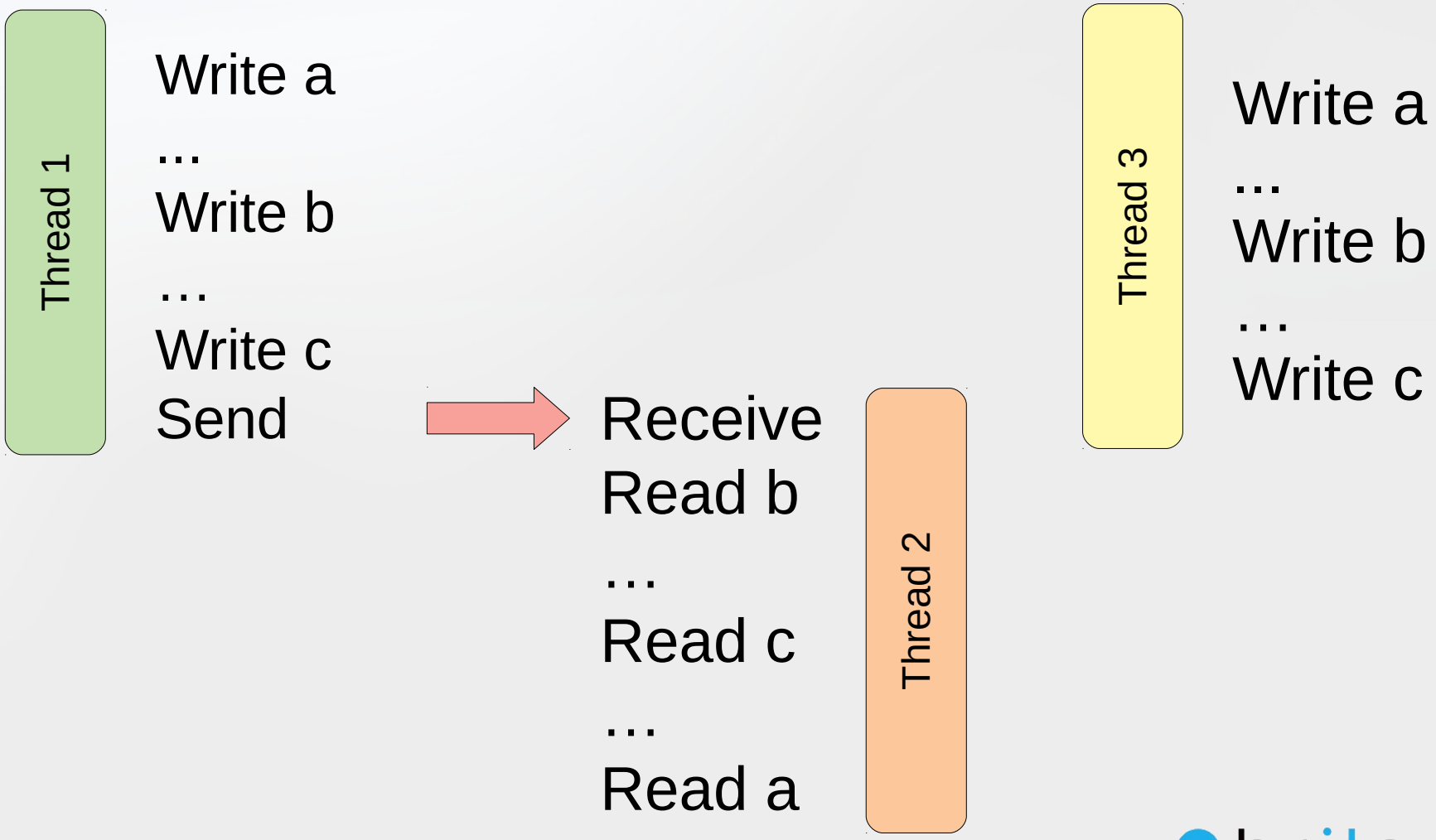
Receive
Read b
...
Read c
...
Read a

Thread 2

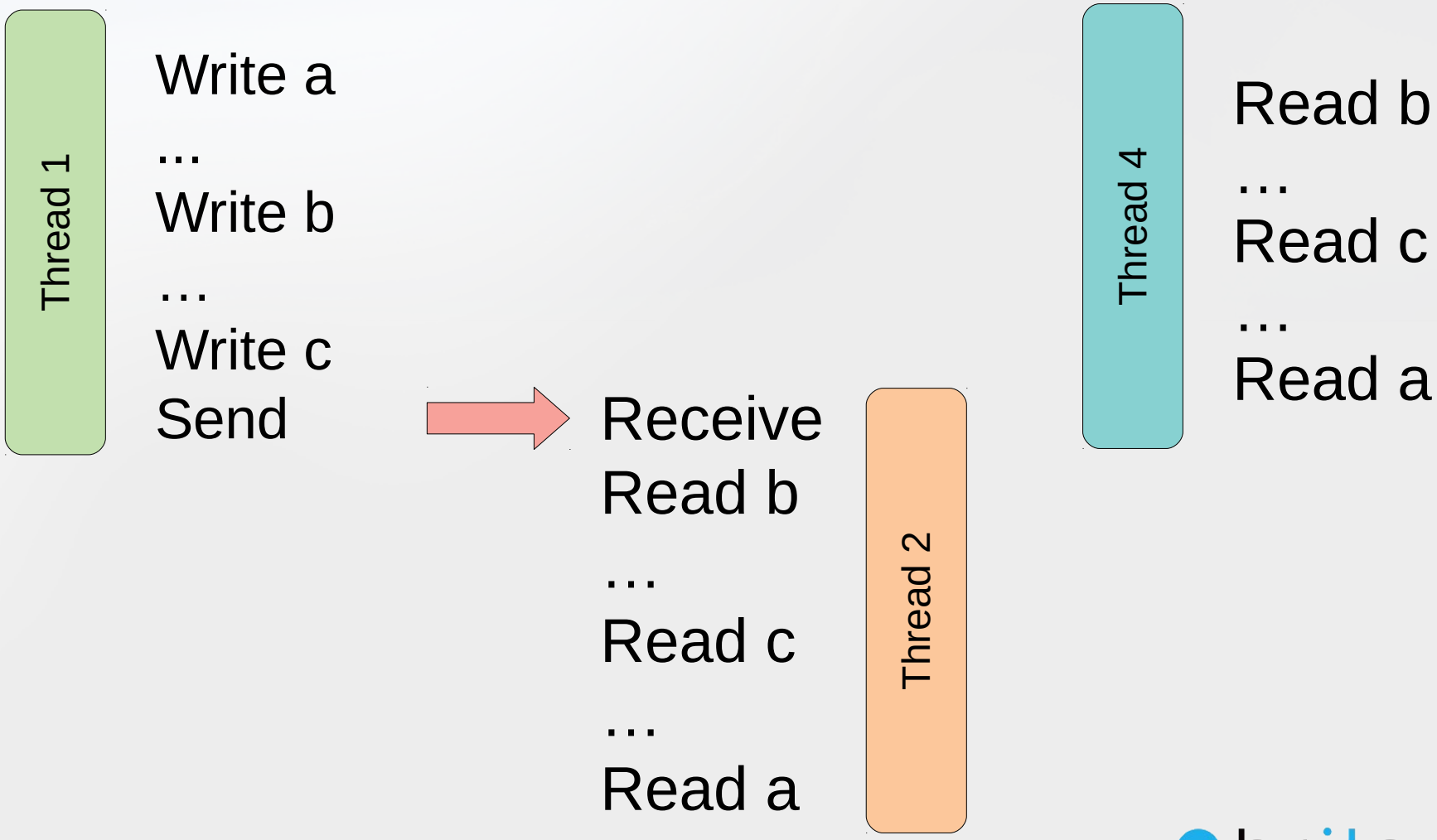
Safe publication

britenet
quality for lease

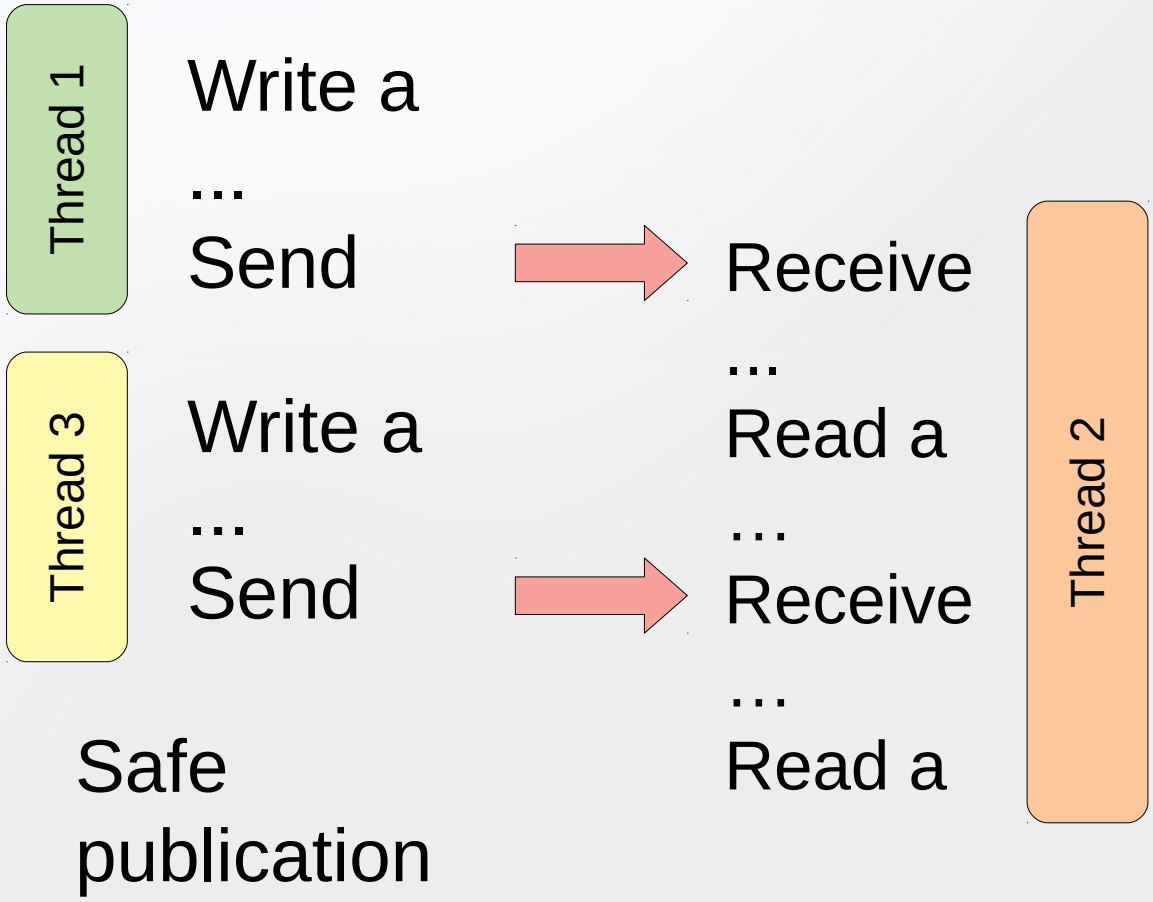
JMM w pigułce – 2 / 2



JMM w pigułce – 2 / 2



JMM w pigułce – 2 / 2



britenet
quality for lease

Double-checked locking

```
private Helper helper;

public Helper getHelper() {
    if (helper == null) {
        synchronized (this) {
            if (helper == null) {
                helper = new Helper();
            }
        }
    }
    return helper;
}
```

Jak zrozumieć JMM?

- Aleksey Shipilëv:
  - YouTube – GeeCON Conference – Java Memory Model Unlearning Experience – 2-5 razy

Jak zrozumieć JMM?

- Aleksey Shipilëv:
  - YouTube – GeeCON Conference – Java Memory Model Unlearning Experience – 2-5 razy
  - YouTube – v JUG – Java Memory Model Pragmatics – 2-3 razy

Jak zrozumieć JMM?

- Aleksey Shipilëv:
  - YouTube – GeeCON Conference – Java Memory Model Unlearning Experience – 2-5 razy
  - YouTube – v JUG – Java Memory Model Pragmatics – 2-3 razy
- Specyfikacja

Jak już musisz...

```
package java.util.concurrent;
package java.util.concurrent.atomic;
package java.util.concurrent.locks;
```

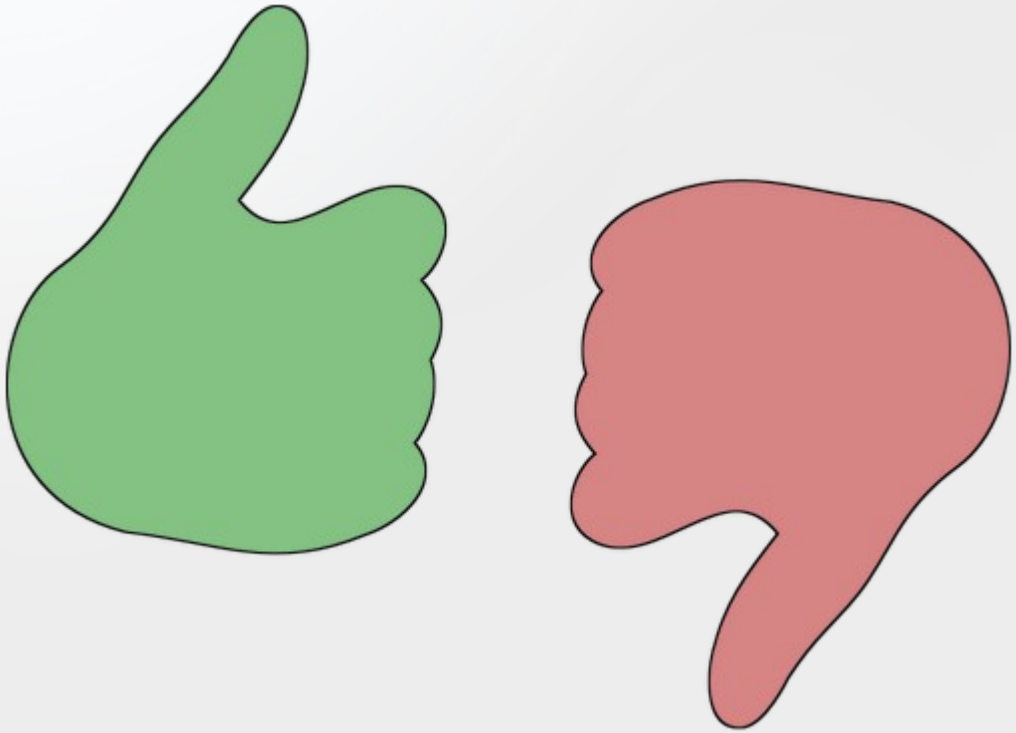
Czy mnie to dotyczy?

britenet
quality for lease

Spring?

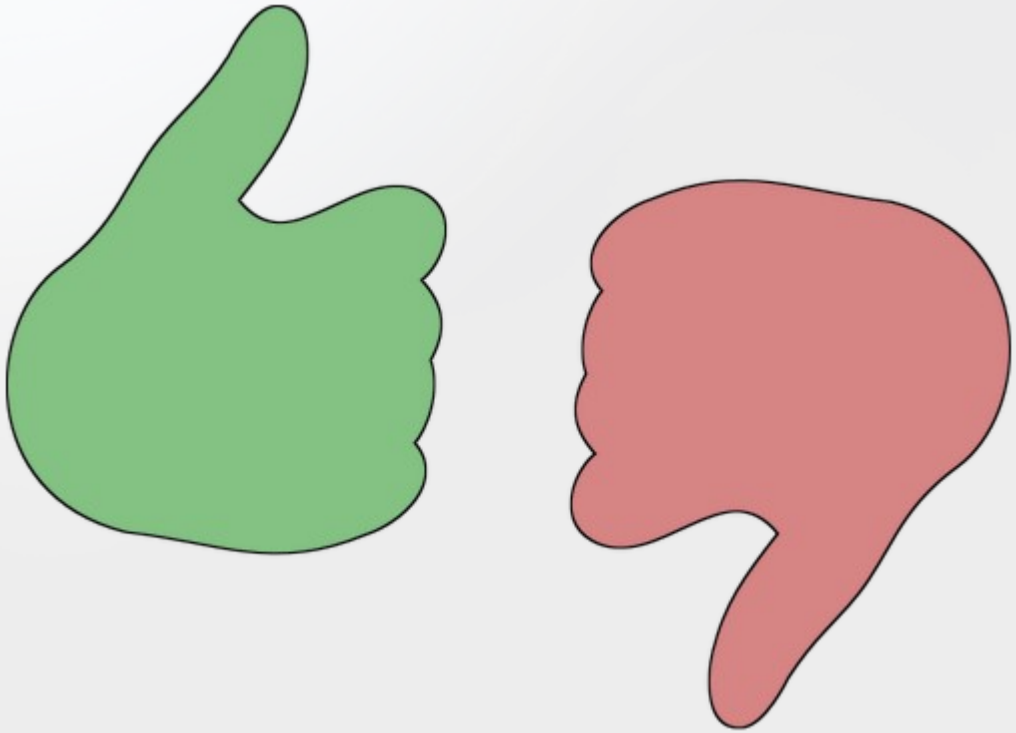
britenet
quality for lease

Spring?



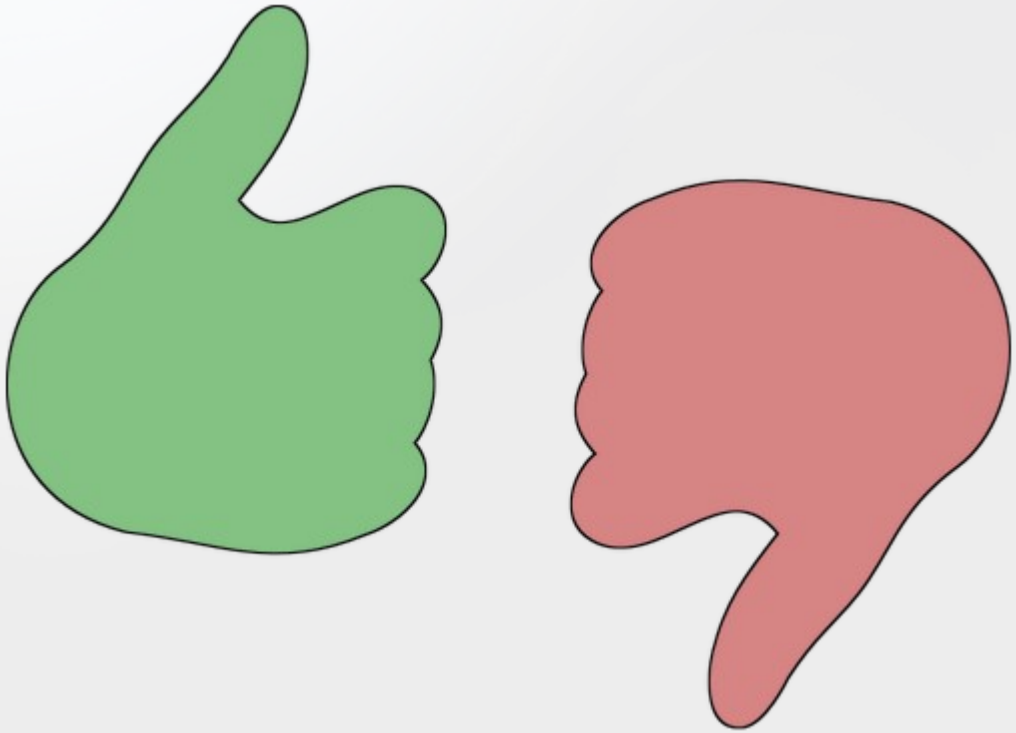
britenet
quality for lease

Spring?



https://jira.spring.io/browse/SPR-4307

Spring?



https://jira.spring.io/browse/SPR-4307
DefaultSingletonBeanRegistry

DefaultSingletonBeanRegistry

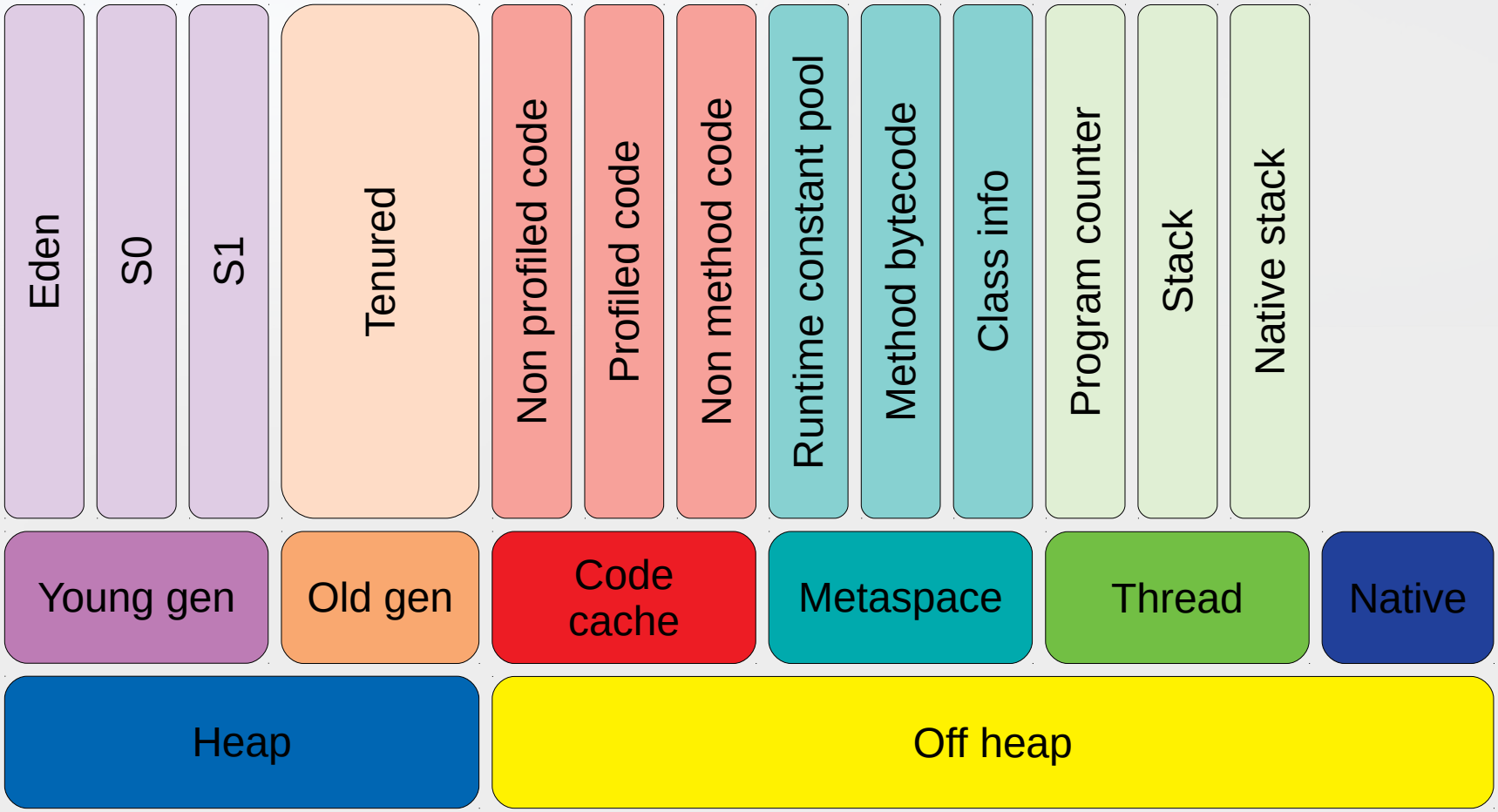
```
public Object getSingleton(String beanName) {
    Object singletonObject =
        this.singletonObjects.get(beanName);
    return (singletonObject != NULL_OBJECT ?
        singletonObject : null);
}
```

https://jira.spring.io/browse/SPR-4672

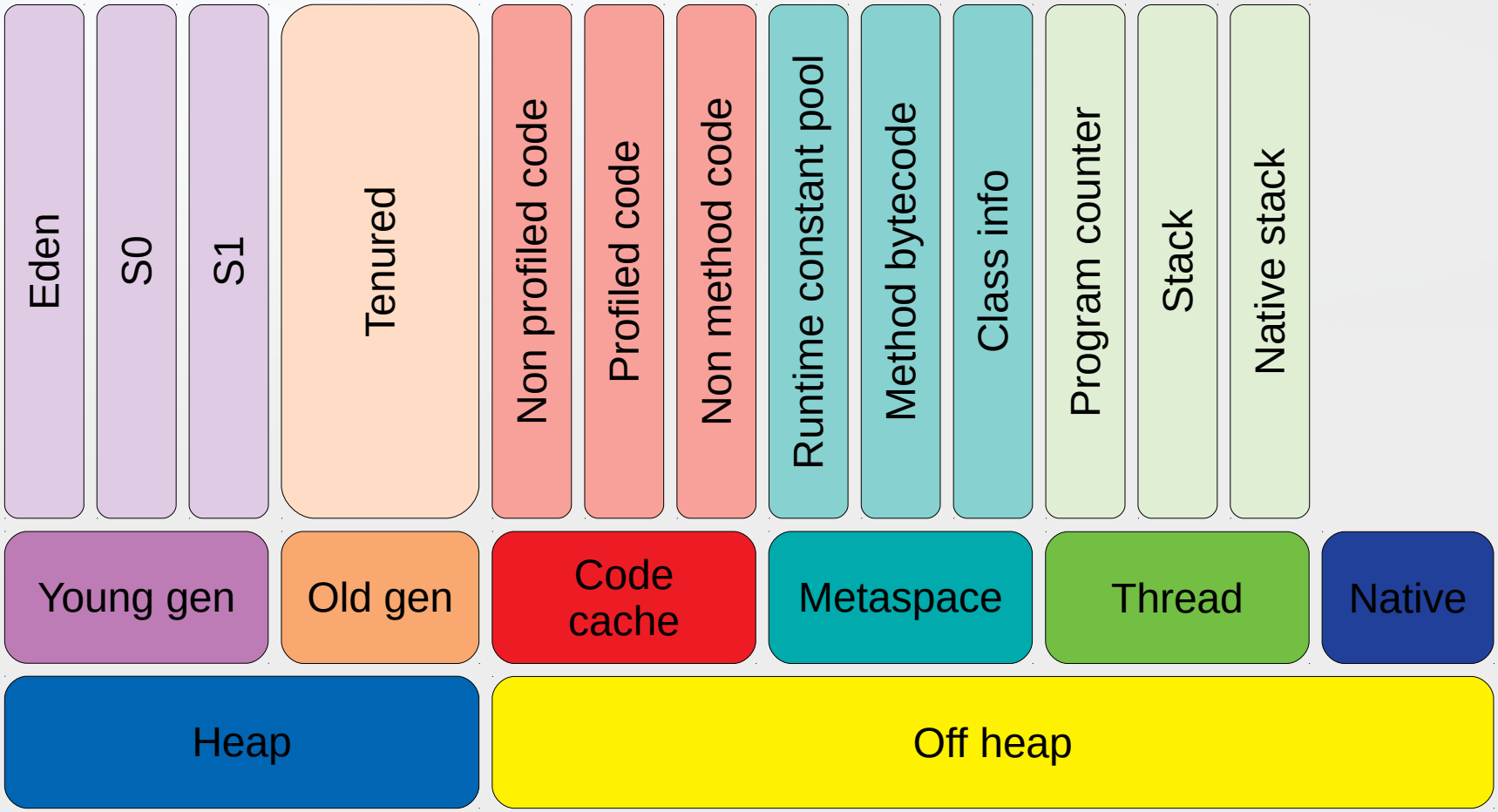
Konsekwencje

- Możliwe fazy stop the world
- Exception cache
- Kod ma znaczenie
- Składnia vs wydajność,
  struktury danych vs wydajność
- Restart JVM --> zaczynamy od nowa
- Concurrency?
- Pamięć

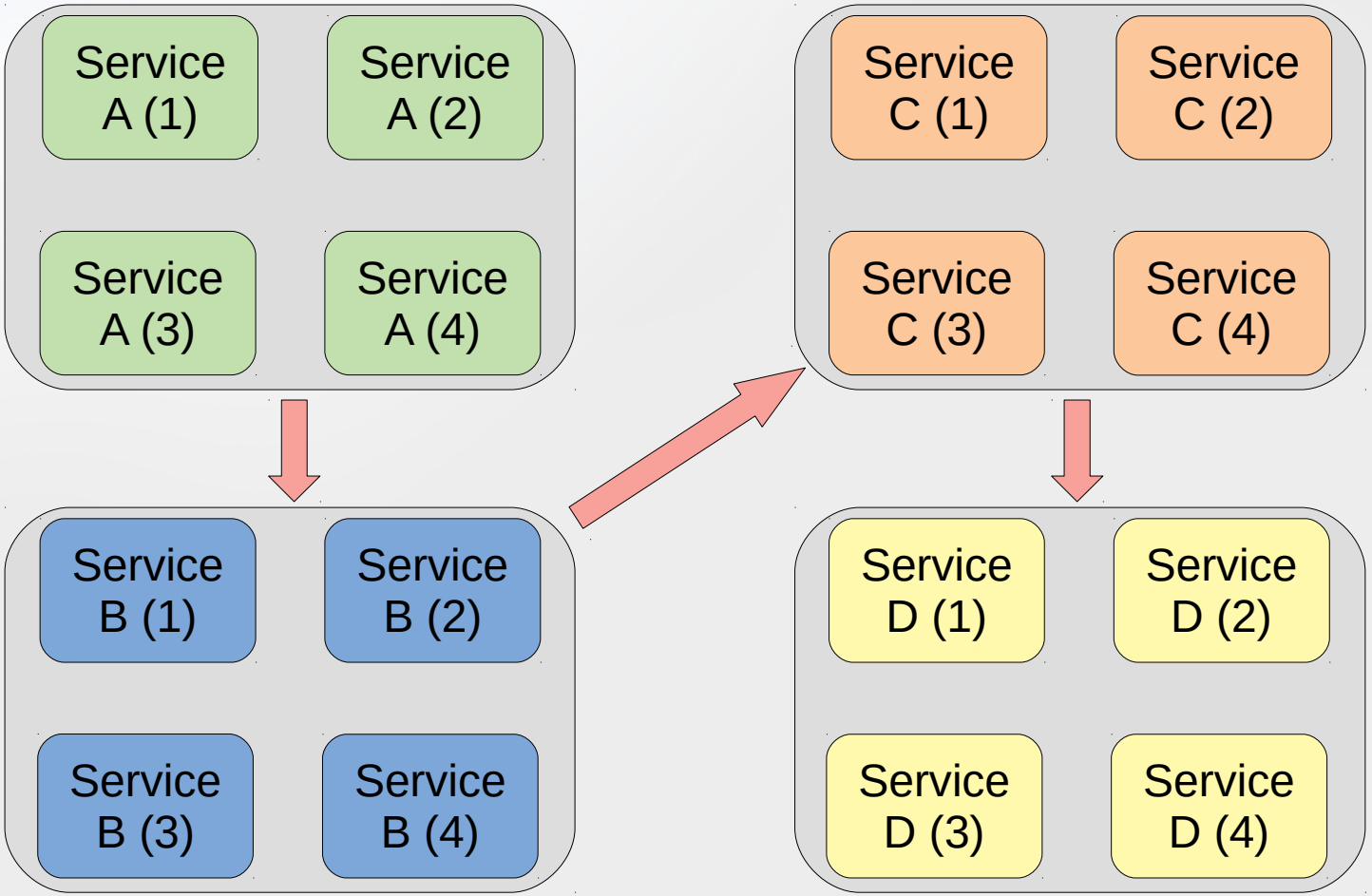
Pamięć w JVM – parallel



JIT w pamięci

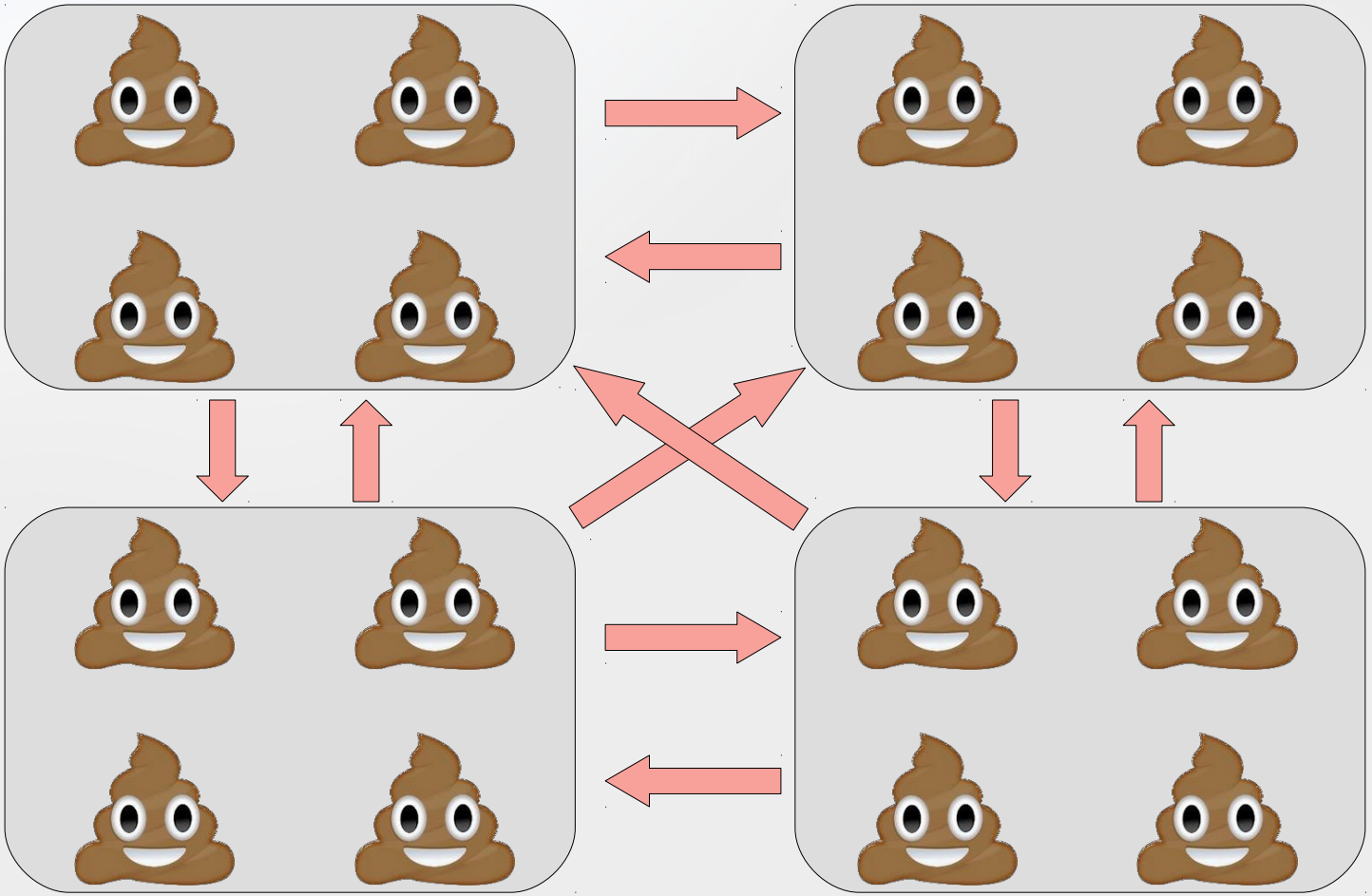


JIT w pamięci – arch. rozp.



britenet
quality for lease

JIT w pamięci – arch. rozp.



britenet
quality for lease

JIT w pamięci – arch. rozp.

Service
...

JIT w pamięci – arch. rozp.

- Max – 240M

Service
...

britenet
quality for lease

JIT w pamięci – arch. rozp.

Service
...

- Max – 240M
- Used – np. 30M

JIT w pamięci – arch. rozp.

Service
...

- Max – 240M
- Used – np. 30M
- Commited – np. 100M

JIT w pamięci – arch. rozp.

Service
...

- Max – 240M
- Used – np. 30M
- Commited – np. 100M
- Wasted – np. 70M

JIT w pamięci – arch. rozp.

Service
...

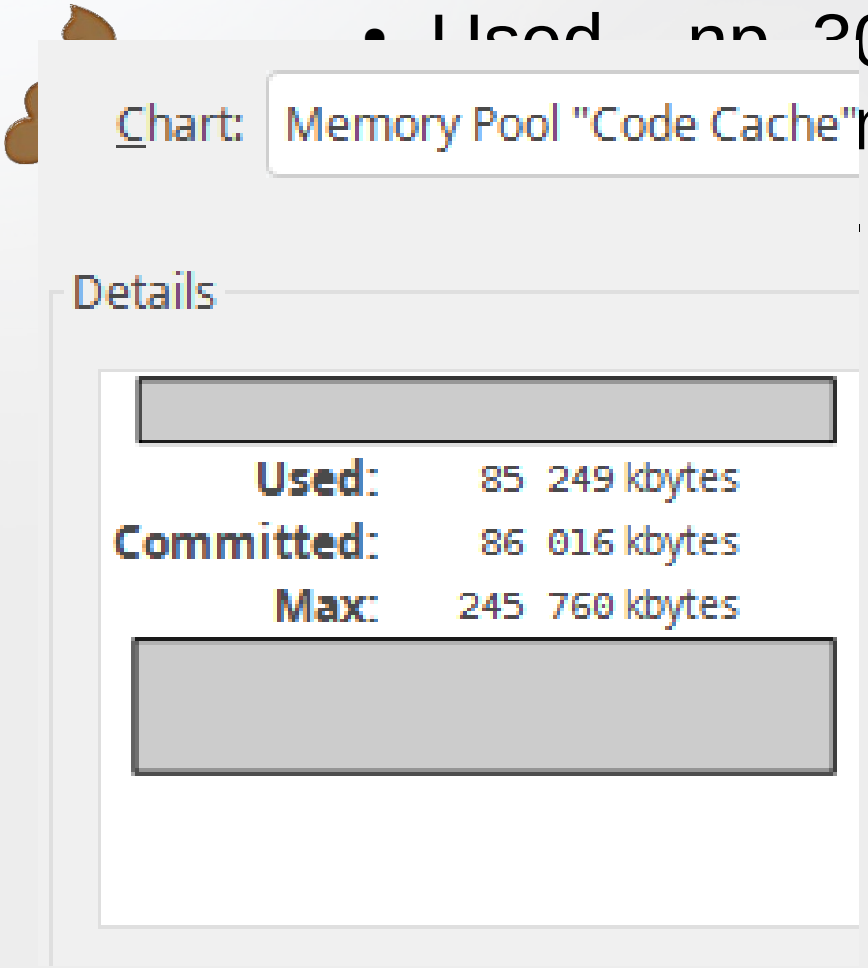
- Max – 240M
- Used – np. 30M
- Commited – np. 100M
- Wasted – np. 70M

np. x1024

JIT w pamięci – arch. rozp.

- Max – 240M
- Used – np. 20M

Service
...



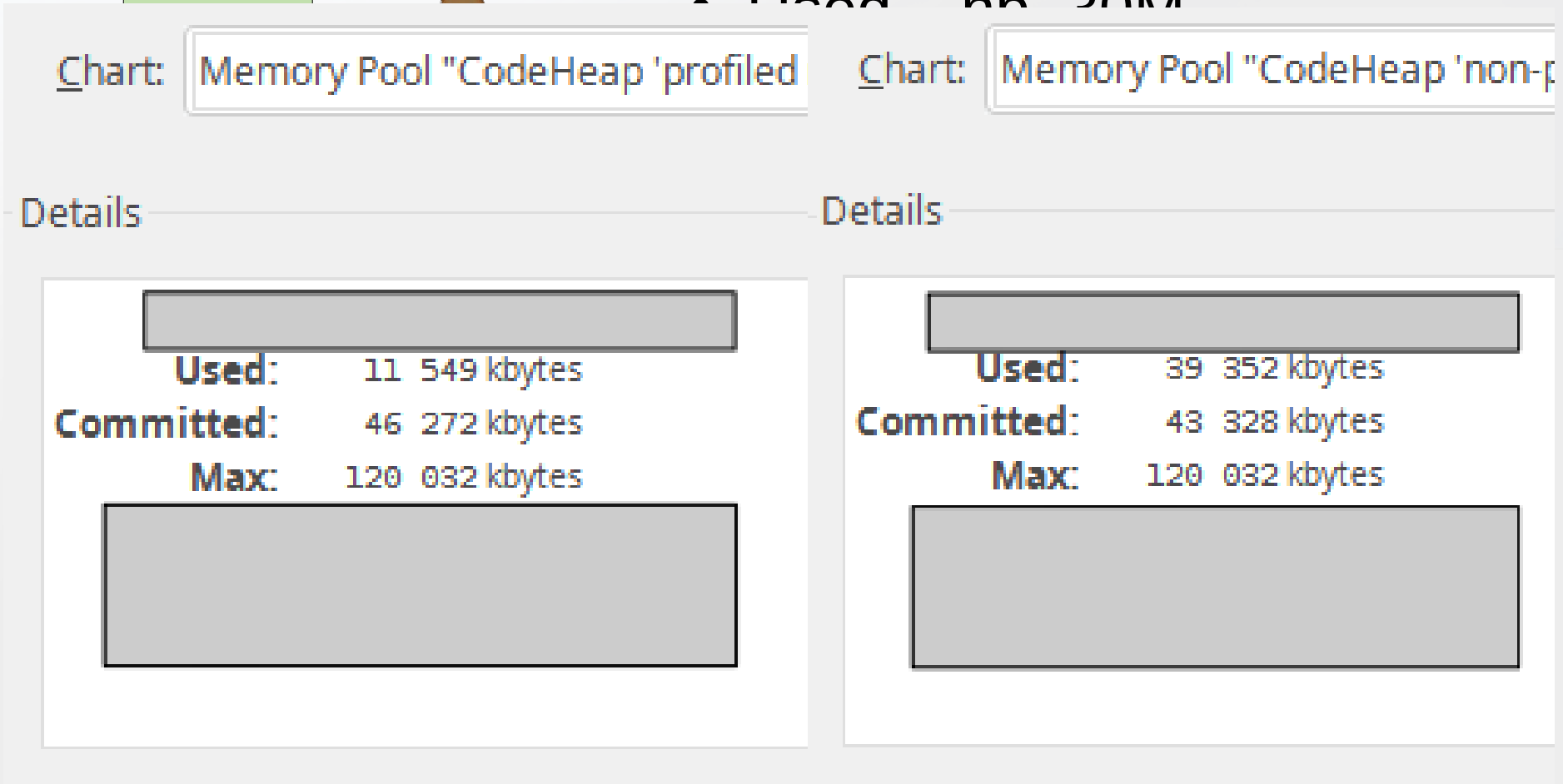
np. 100M

70M

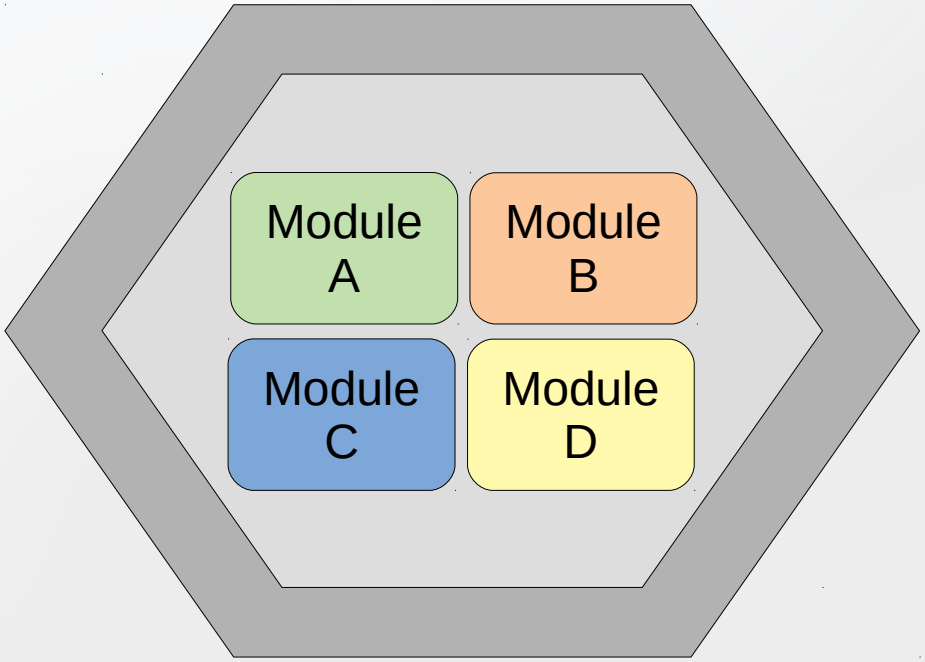
britenet
quality for lease

JIT w pamięci – arch. rozp.

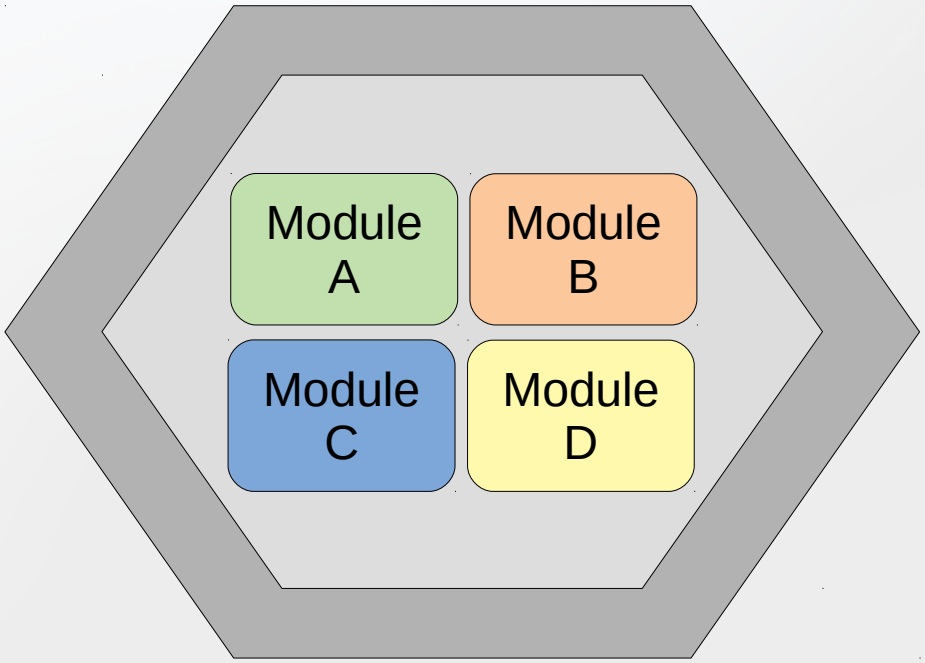
- Max – 240M
- Used – np. 20M



JIT w pamięci – monolit



JIT w pamięci – monolit



britenet
quality for lease

Może coś da się zrobić?

britenet
quality for lease

Może coś da się zrobić?

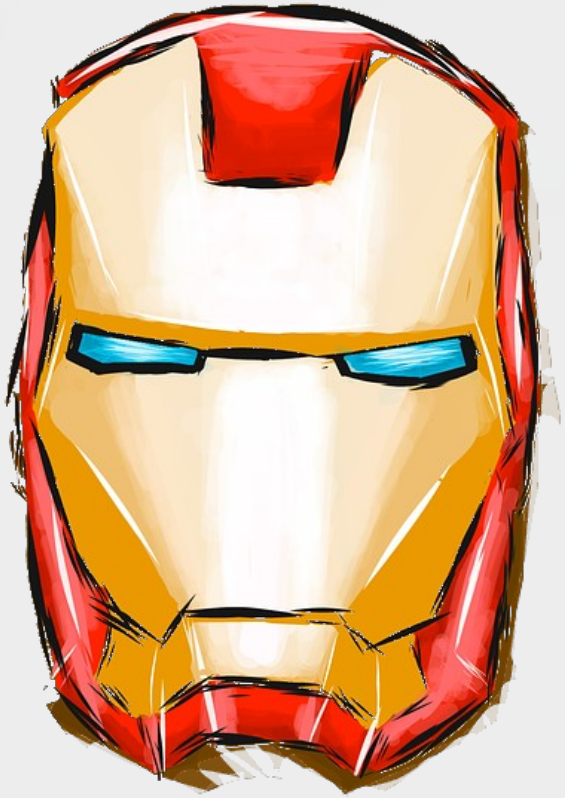
Kompilator

britenet
quality for lease

Może coś da się zrobić?

Kompilator

Sweeper



britenet
quality for lease

Może coś da się zrobić?

Kompilator

Sweeper

1/140000605

britenet
quality for lease

Sweeper – success story

- Charakterystyka użycia kodu zmienna w czasie

Sweeper – success story

- Charakterystyka użycia kodu zmienna w czasie
  - Start aplikacji / późniejsze użycie

Sweeper – success story

- Charakterystyka użycia kodu zmienna w czasie
  - Start aplikacji / późniejsze użycie
  - Inne użycie aplikacji w czasie

Sweeper – success story

- Charakterystyka użycia kodu zmienna w czasie
  - Start aplikacji / późniejsze użycie
  - Inne użycie aplikacji w czasie
    - np. w ciągu dnia biznes, wieczorem klienci indywidualni

Sweeper – success story

- Charakterystyka użycia kodu zmienna w czasie
  - Start aplikacji / późniejsze użycie
  - Inne użycie aplikacji w czasie
    - np. w ciągu dnia biznes, wieczorem klienci indywidualni
    - np. w ciągu dni aplikacja webowa, w nocy przetwarzanie batchowe

Konsekwencje

- Możliwe fazy stop the world
- Exception cache
- Kod ma znaczenie
- Składnia vs wydajność,
  struktury danych vs wydajność
- Restart JVM --> zaczynamy od nowa
- Concurrency?
- Pamięć
- Nowe zachowanie --> spowolnienie aplikacji

Konsekwencje

- Możliwe fazy stop the world
- Exception cache
- Kod ma znaczenie
- Składnia vs wydajność,
  struktury danych vs wydajność
- Restart JVM --> zaczynamy od nowa
- Concurrency?
- Pamięć
- Nowe zachowanie --> spowolnienie aplikacji
- Testy wydajnościowe

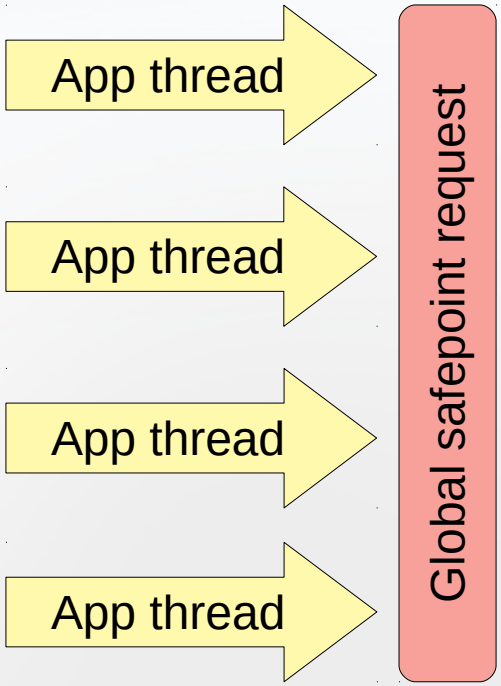
Konsekwencje – mniej ważne

britenet
quality for lease

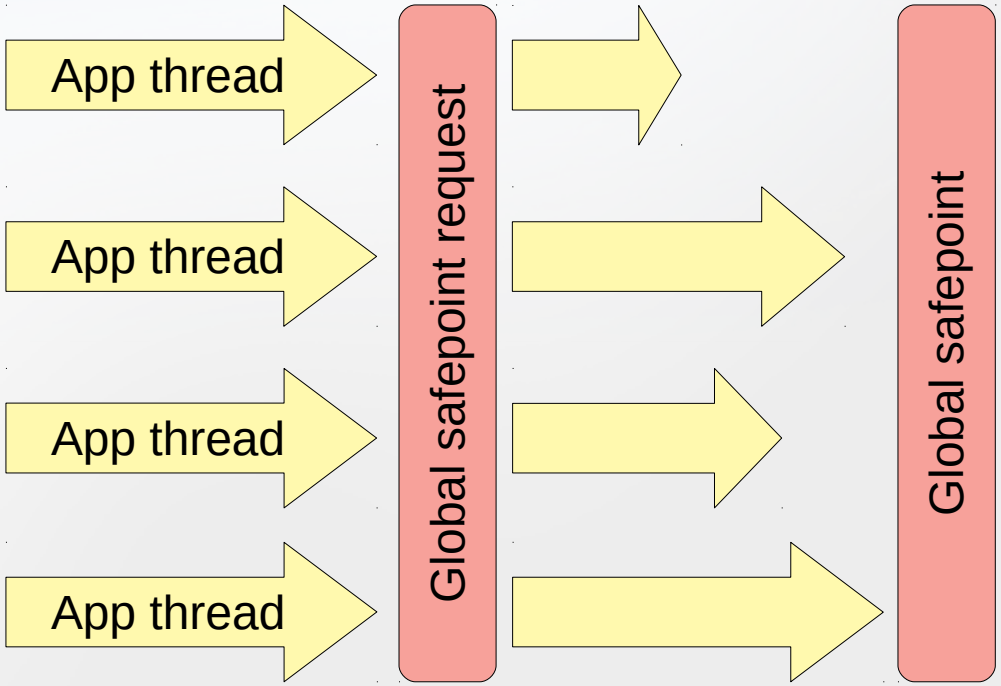
Konsekwencje – mniej ważne

Intrinsics...

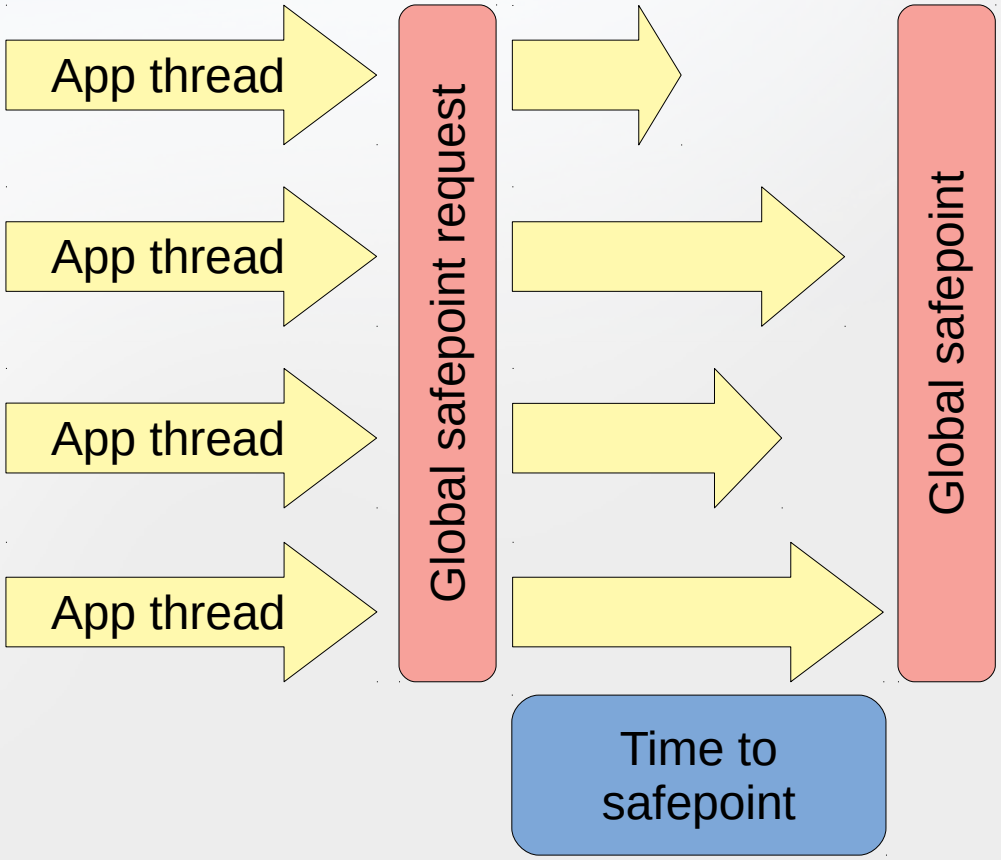
Mniejsza liczba safepointów



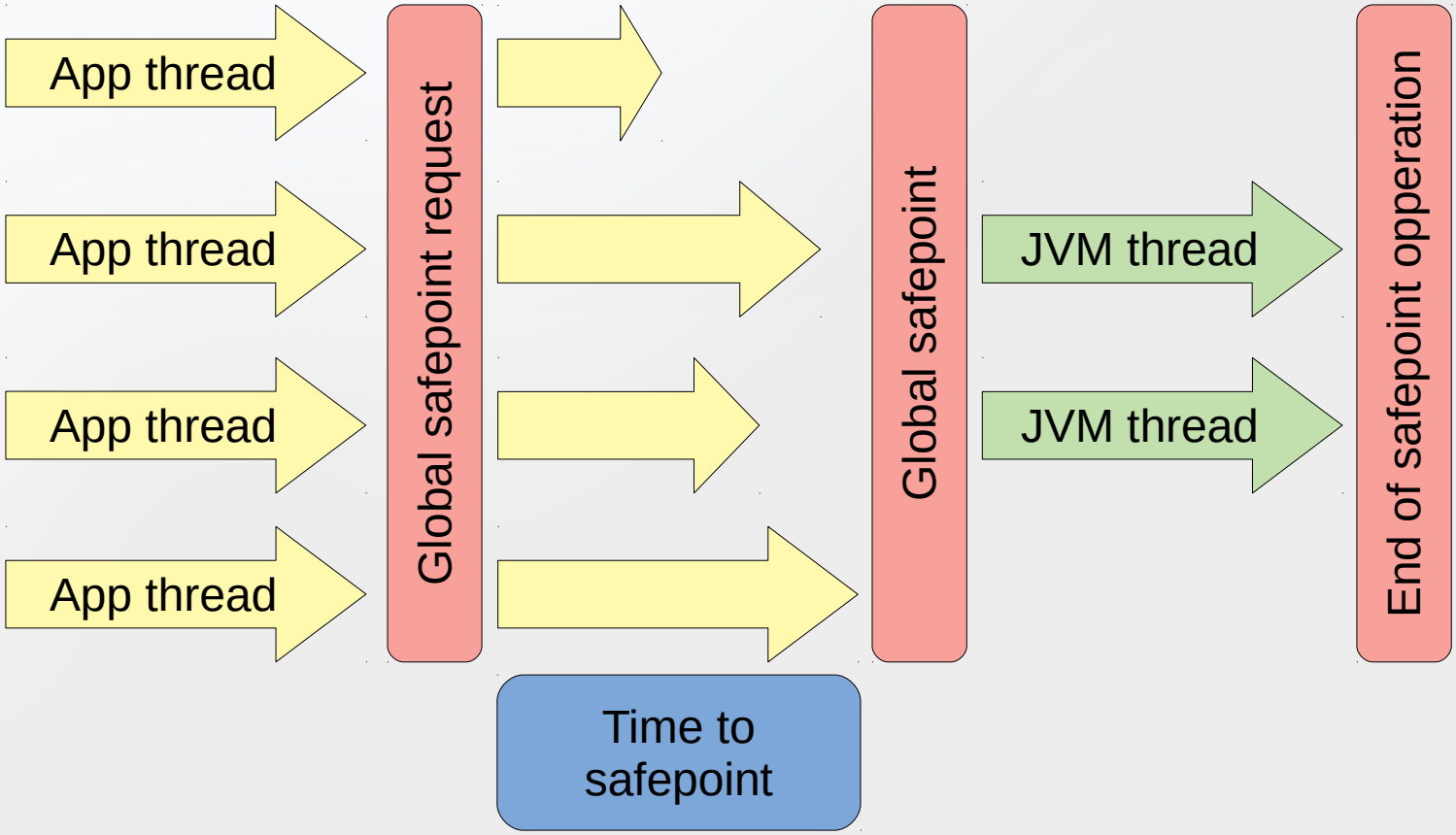
Mniejsza liczba safepointów



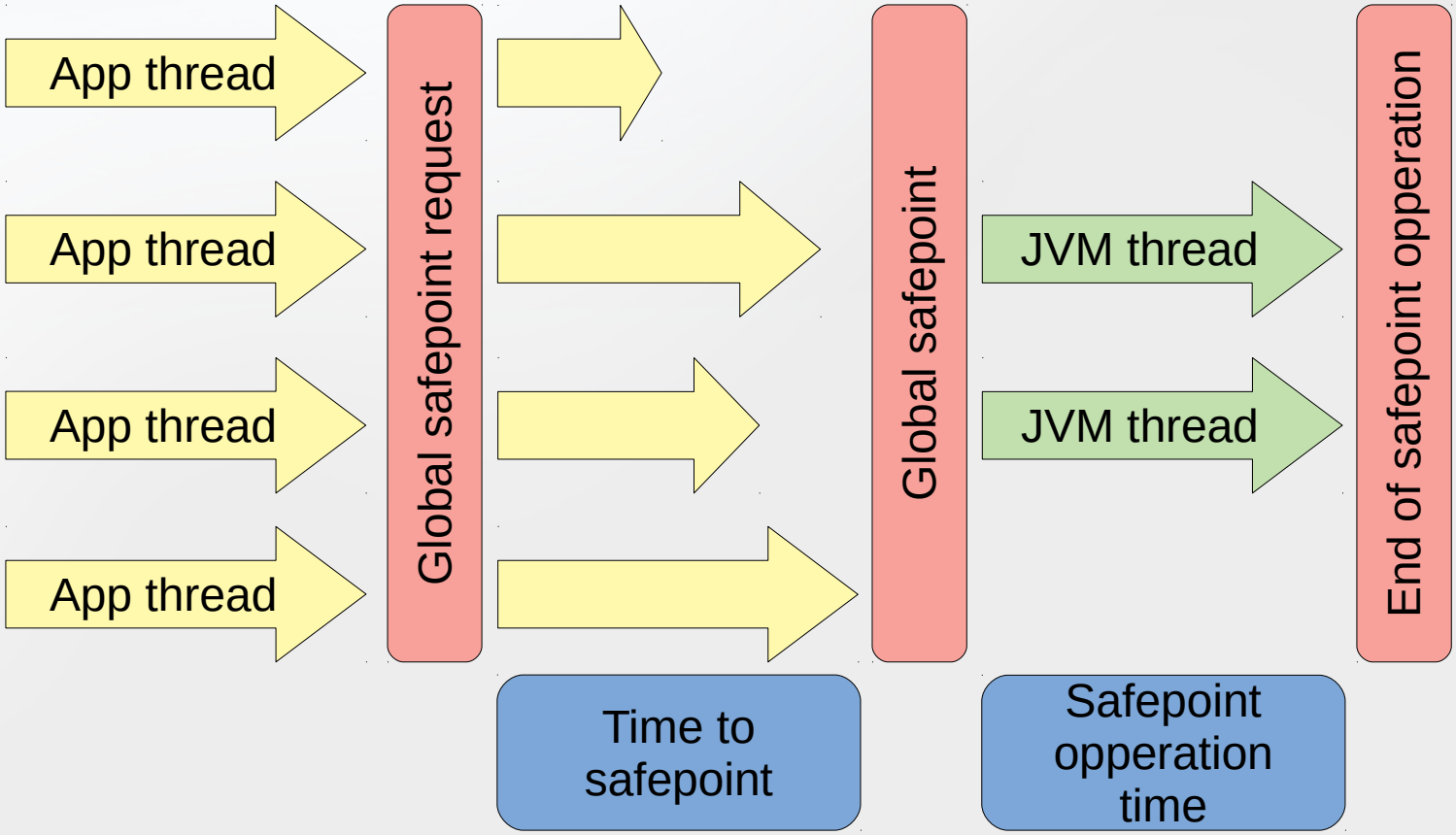
Mniejsza liczba safepointów



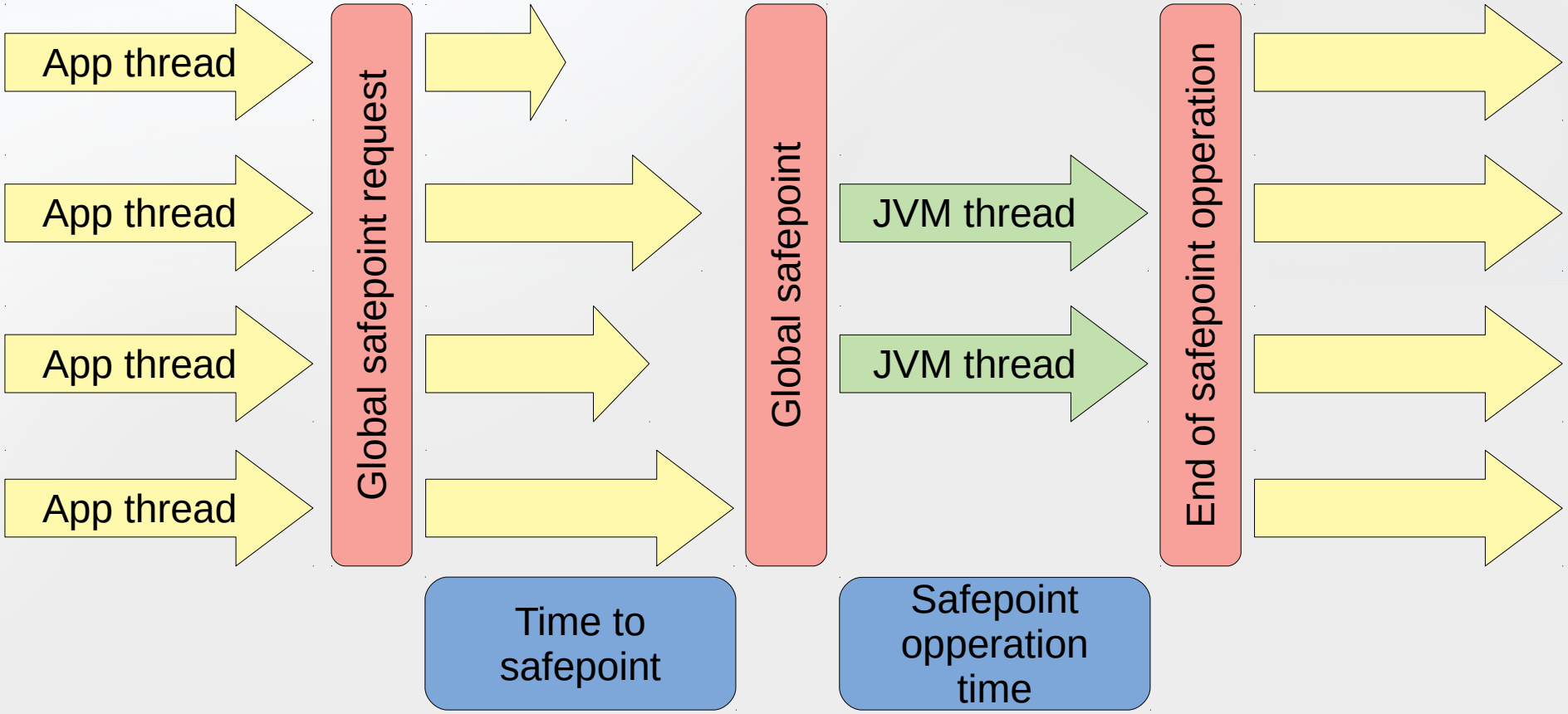
Mniejsza liczba safepointów



Mniejsza liczba safepointów



Mniejsza liczba safepointów



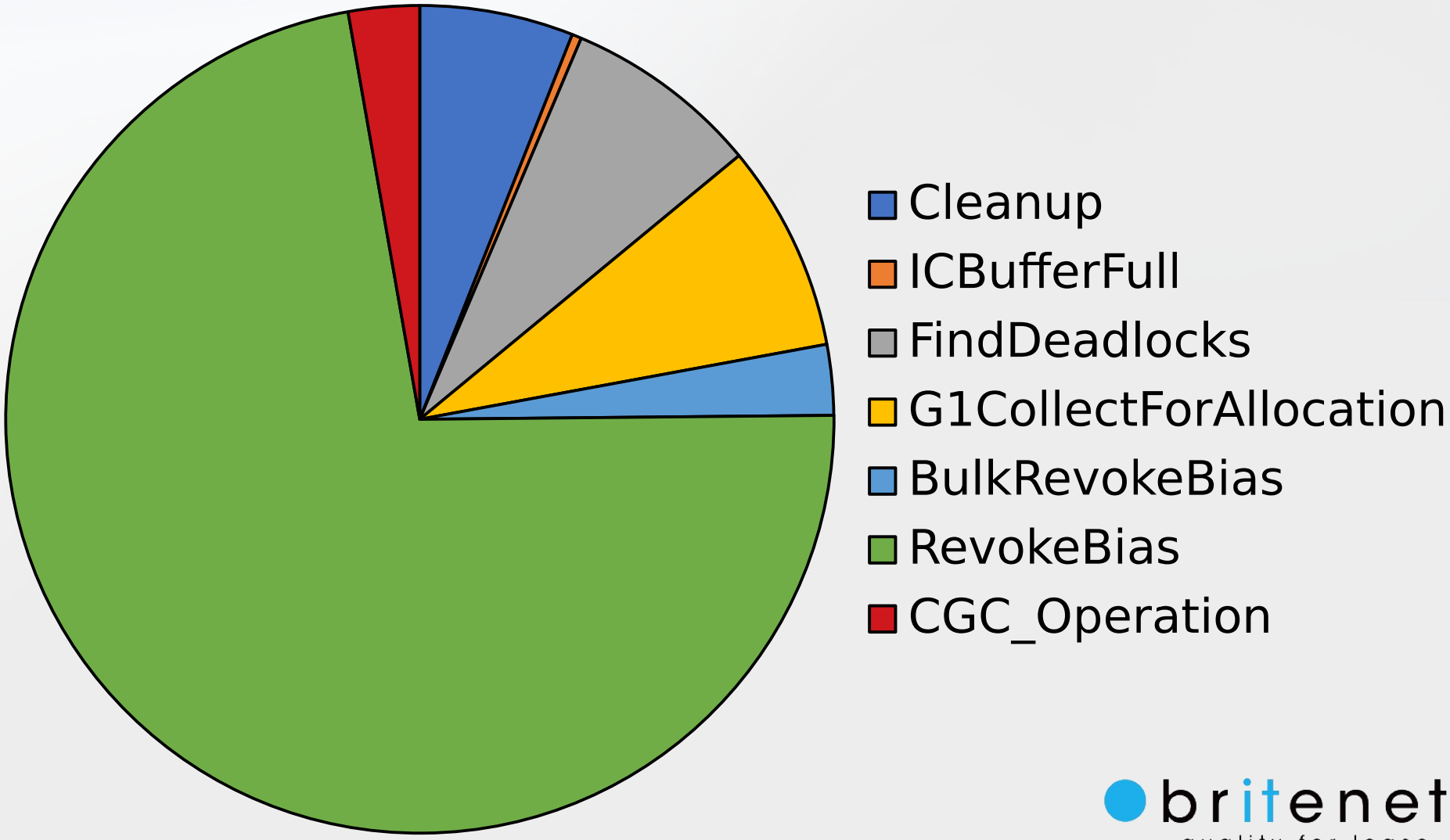
britenet
quality for lease

Safepointy – praktyka

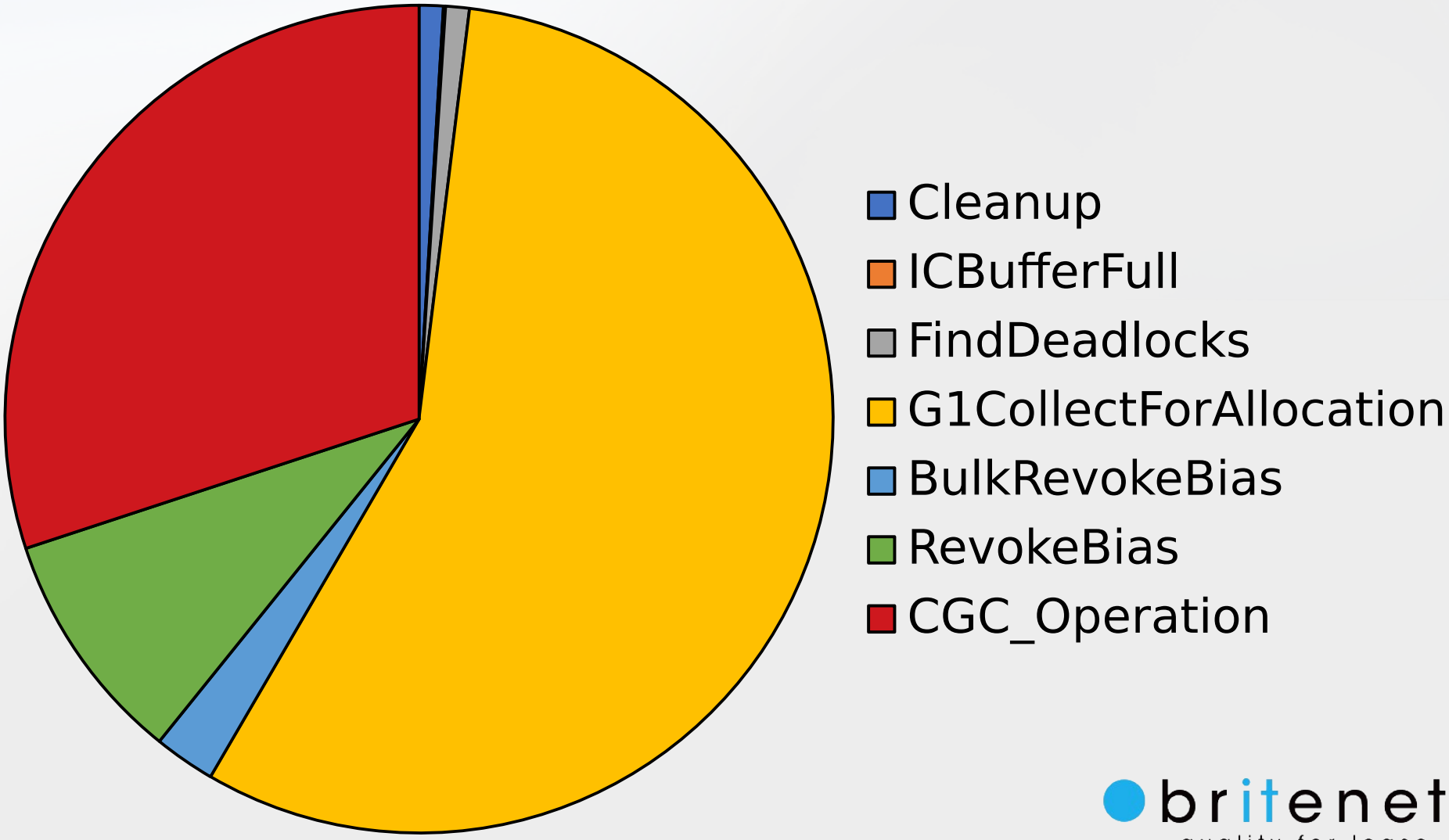
- 37 668 safepoint operations / 2h

Percentile	Time to safepoint	Operation time	Application time
50	0,04	0,21	**1,97**
75	0,05	0,31	262,60
99	0,15	34,89	1 000,17
99,9	1,82	72,36	1 151,64
99,99	7,48	93,44	2 031,07
99,999	10,05	110,73	3 032,11
100	**10,05**	110,73	**3 032,11**
Average	**0,05**	2,04	**189,03**

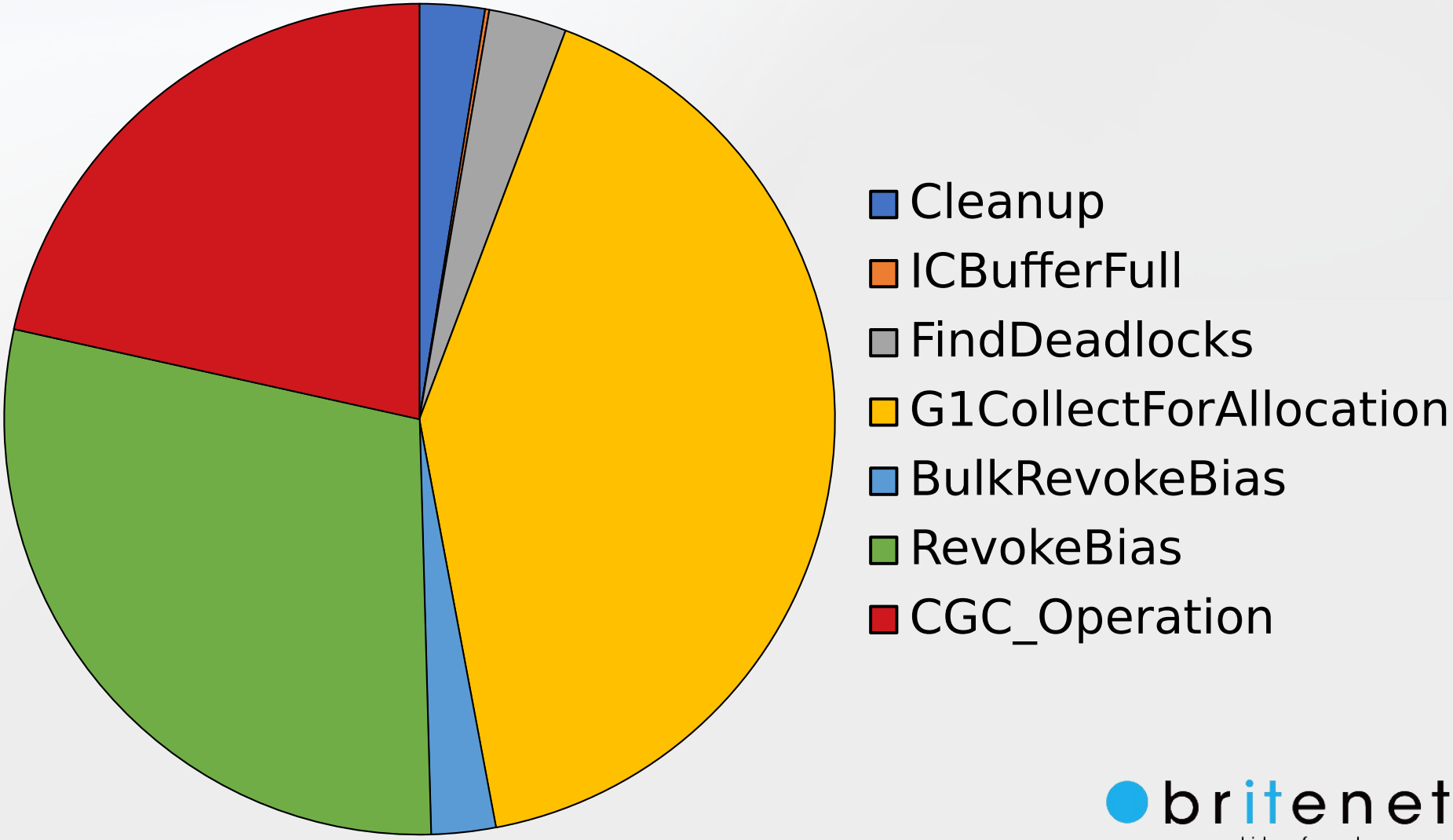
Safepointy – liczba



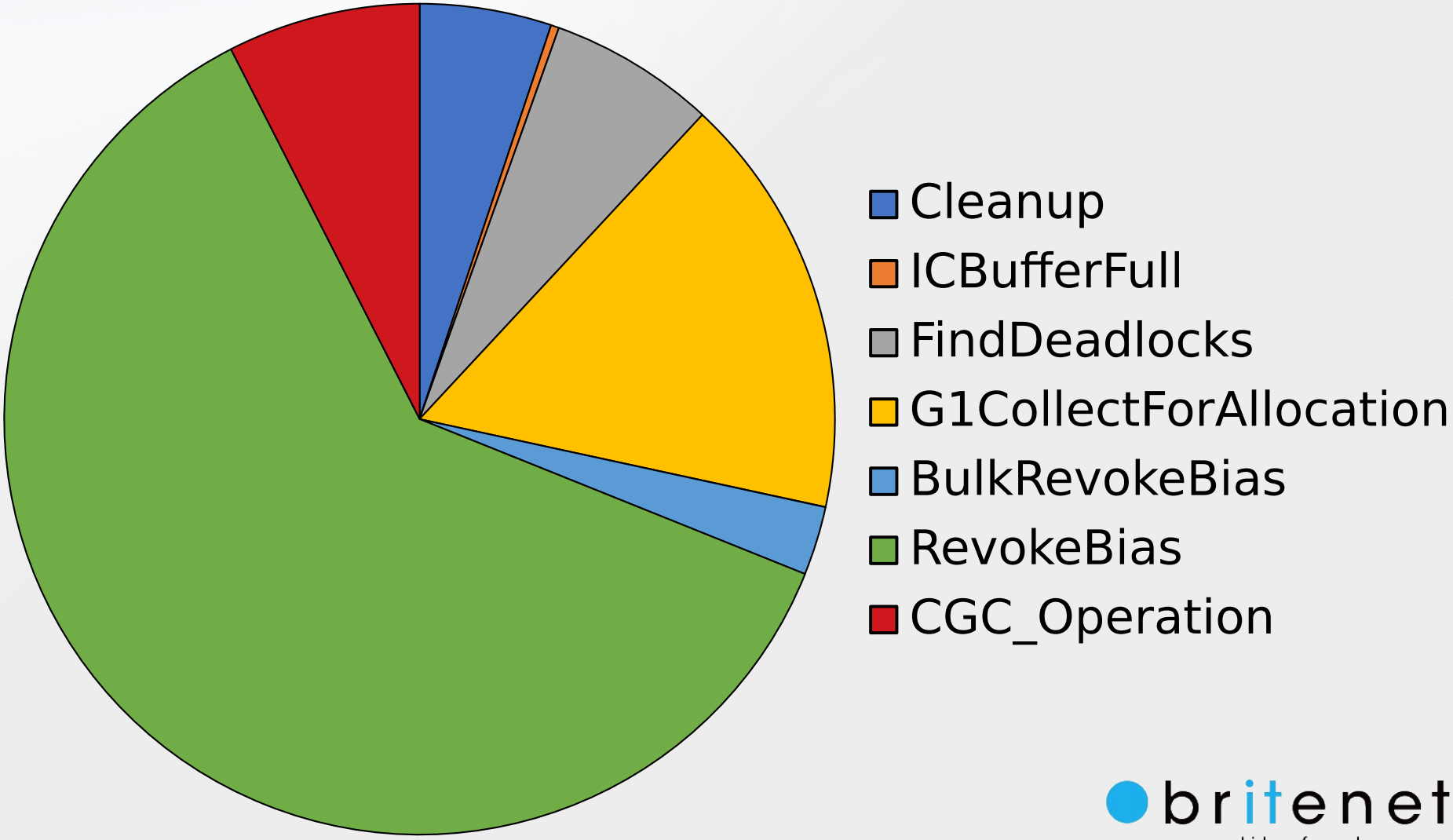
Safepointy – total time + TTS



Safepointy – total time + 1ms



Safepointy – total time + 10ms



Czy ktoś coś z tym robi?

britenet
quality for lease

Czy ktoś coś z tym robi?

britenet
quality for lease

Czy ktoś coś z tym robi?

- ShenandoahGC i ZGC

Czy ktoś coś z tym robi?

- ShenandoahGC i ZGC
- Np. optymalizacja „loop strip mining” – JDK 10 – z bugiem :) Bug id: 8220374

Czy ktoś coś z tym robi?

- ShenandoahGC i ZGC
- Np. optymalizacja „loop strip mining” – JDK 10 – z bugiem :) Bug id: 8220374
- Naprawione w JDK 13 (b13) i downport do JDK 12 (u2) i JDK 11 (u4)

Czy ktoś coś z tym robi?

- ShenandoahGC i ZGC
- Np. optymalizacja „loop strip mining” – JDK 10 – z bugiem :) Bug id: 8220374
- Naprawione w JDK 13 (b13) i downport do JDK 12 (u2) i JDK 11 (u4)
- Tylko dla ZGC, ShenandoahGC i G1

Złote rady wujka Krzyśka

- Monitoruj Code cache

Złote rady wujka Krzyśka

- Monitoruj Code cache
- Używaj prymitywów

Złote rady wujka Krzyśka

- Monitoruj Code cache
- Używaj prymitywów
- Zainteresuj się wydajnością aktualnie pisanego kodu

Złote rady wujka Krzyśka

- Monitoruj Code cache
- Używaj prymitywów
- Zainteresuj się wydajnością aktualnie pisanego kodu
- Używaj nowych wersji JDK

Złote rady wujka Krzyśka

- Monitoruj Code cache
- Używaj prymitywów
- Zainteresuj się wydajnością aktualnie pisanego kodu
- Używaj nowych wersji JDK
- Używaj metod synchronizacji jak najwyższego poziomu

Złote rady wujka Krzyśka

- Monitoruj Code cache
- Używaj prymitywów
- Zainteresuj się wydajnością aktualnie pisanego kodu
- Używaj nowych wersji JDK
- Używaj metod synchronizacji jak najwyższego poziomu
- Koduj zgodnie ze specyfikacją, nie implementacją

Złote rady wujka Krzyśka

- Monitoruj Code cache
- Używaj prymitywów
- Zainteresuj się wydajnością aktualnie pisanego kodu
- Używaj nowych wersji JDK
- Używaj metod synchronizacji jak najwyższego poziomu
- Koduj zgodnie ze specyfikacją, nie implementacją
- Nie rób optymalizacji które robi JIT

Złote rady wujka Krzyśka

- Monitoruj Code cache
- Używaj prymitywów
- Zainteresuj się wydajnością aktualnie pisanego kodu
- Używaj nowych wersji JDK
- Używaj metod synchronizacji jak najwyższego poziomu
- Koduj zgodnie ze specyfikacją, nie implementacją
- Nie rób optymalizacji które robi JIT
- Nie traktuj kluczowych składowych systemu jako BlackBox

Skills

Hard	Soft

britenet
quality for lease

Skills

Hard	Soft	
	CQRS	Event sourcing
	DDD	TDD/BDD
	Hexagonal architecture	Microservices
	Hateoas	Restful
	Machine learning	Pipes and filters
	...	

britenet
quality for lease

Skills

Hard		Soft	
JIT	Garbage collector	CQRS	Event sourcing
Classloader	Safepoint	DDD	TDD/BDD
Spring proxy	Hibernate caches	Hexagonal architecture	Microservices
Bytecode	Memory layout	Hateoas	Restful
JMM	Off heap	Machine learning	Pipes and filters
...		...	

Skills

Hard		Soft	
JIT	Garbage collector	CQRS	Event sourcing
Classloader	Safep	D	TDD/BDD
Spring proxy	Hibernate h	nal ecture	Microservices
Bytecode	Memory ut	oas	Restful
JMM	Off heap	Machine learning	Pipes and filters
...		...	

britenet
quality for lease

Dziękuję

britenet
quality for lease

conf@kś.pl
jug2019@kś.pl

Pytania?

britenet
quality for lease