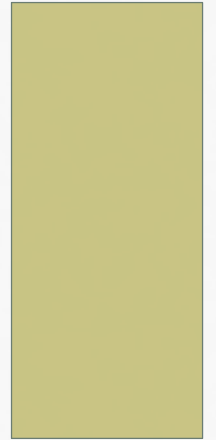


Java Memory Model



Happens-before

В Java Memory Model введена такая абстракция как happens-before. Она обозначает, что если операция X связана отношением happens-before с операцией Y, то весь код следуемый за операцией Y, выполняемый в одном потоке, видит все изменения, сделанные другим потоком, до операции X.

Связь happens-before транзитивна, т.е. если X happens-before Y, а Y happens-before Z, то X happens-before Z.

Happens-before

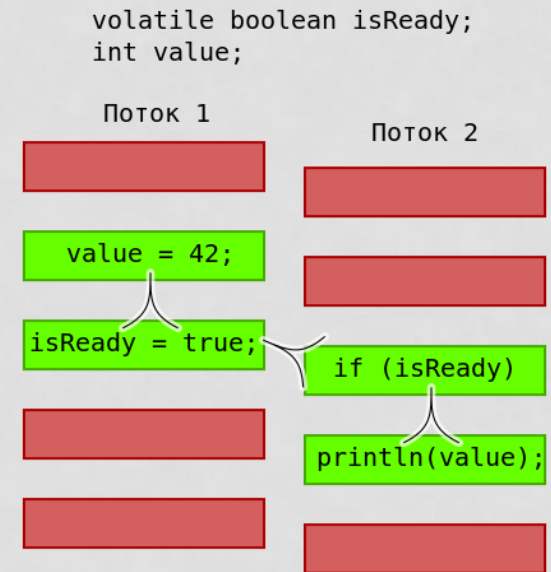
Синхронизация и мониторы:

- Захват монитора (начало `synchronized`, метод `lock`) и всё, что после него в том же потоке.
- Возврат монитора (конец `synchronized`, метод `unlock`) и всё, что перед ним в том же потоке.
 - Таким образом, оптимизатор может заносить строки в синхроблок, но не наружу.
- Возврат монитора и последующий захват другим потоком.

Happens-before

Запись и чтение:

- Любые зависимости по данным (то есть запись в любую переменную и последующее чтение её же) в одном потоке.
- В одном потоке перед записью в `volatile`-переменную, и сама запись.
- `volatile`-чтение и всё, что после него в том же потоке.
- Запись в `volatile`-переменную и последующее считывание её же.
 - Для объектных переменных (например, `volatile List x`;) столь сильные гарантии выполняются для ссылки на объект, но не для его содержимого.



Happens-before

Обслуживание объекта:

- Статическая инициализация и любые действия с любыми экземплярами объектов.
- Запись в `final`-поля в конструкторе и всё, что после конструктора.
 - Как исключение из всеобщей транзитивности, это соотношение `happens-before` не соединяется транзитивно с другими правилами и поэтому может вызвать межпоточную гонку.
- Любая работа с объектом и `finalize()`.

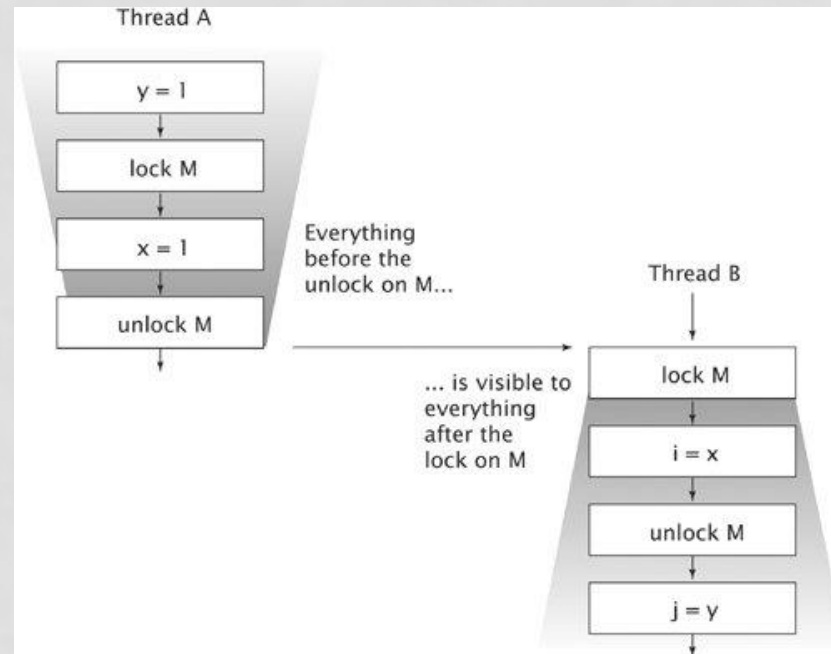
Happens-before

Обслуживание потока:

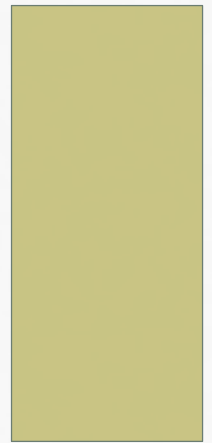
- Запуск потока и любой код в потоке.
- Зануление переменных, относящихся к потоку, и любой код в потоке.
- Код в потоке и `join()`; код в потоке и `isAlive() == false`.
- `interrupt()` потока и обнаружение факта останова.

Happens-before

В отношении happens-before есть очень большой дополнительный бонус: данное отношение дает не только видимость `volatile` полей или результатов операций защищенных монитором или локом, но и видимость вообще всего, что делалось до события happens-before.



Synchronized VS volatile



synchronized

```
class B<T> {
    T x;

    public void set(T v) {
        synchronized (this) {
            x = v;
        } // "release" on unlock
    }

    public T get() {
        synchronized (this) { // "acquire" on lock
            return x;
        }
    }
}
```

volatile

```
class B<T> {  
    volatile T x;  
  
    public void set(T v) {  
        x = v; // "release" on volatile store  
    }  
  
    public T get() {  
        return x; // "acquire" on volatile load  
    }  
}
```

Соответственно

```
int a, b, x;
```

```
synchronized (lock) {  
    a = x;  
    b = 1;  
}
```

```
int x;
```

```
volatile boolean busyFlag;
```

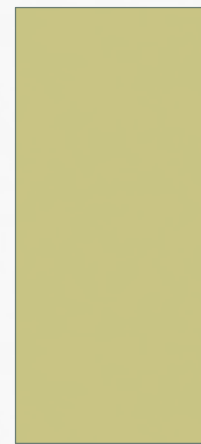
```
while (!compareAndSet(busyFlag, false, true));
```

```
a = x;
```

```
b = 1;
```

```
busyFlag = false;
```

volatile => можно не
думать?



```
@JCStressTest
@State
public class VolatileCounters {
    volatile int x;

    @Actor
    void actor1() {
        for (int i = 0; i < 10; i++) {
            x++;
        }
    }

    @Actor
    void actor2() {
        for (int i = 0; i < 10; i++) {
            x++;
        }
    }

    @Arbiter
    public void arbiter(IntResult1 r) {
        r.r1 = x;
    }
}
```

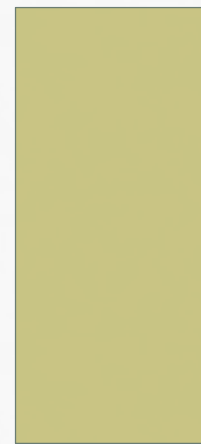
Магия

(fork: #1, iteration #1, JVM args: [-server])

Observed state Occurrences Expectation Interpretation

10	153,217	ACCEPTABLE	\$x \$y
11	273,440	ACCEPTABLE	\$x \$y
12	465,262	ACCEPTABLE	\$x \$y
13	611,123	ACCEPTABLE	\$x \$y
14	810,790	ACCEPTABLE	\$x \$y
15	1,139,737	ACCEPTABLE	\$x \$y
16	1,189,164	ACCEPTABLE	\$x \$y
17	1,163,565	ACCEPTABLE	\$x \$y
18	1,149,772	ACCEPTABLE	\$x \$y
19	986,010	ACCEPTABLE	\$x \$y
20	7,449,917	ACCEPTABLE	\$x \$y
6	4	ACCEPTABLE	\$x \$y
7	6,442	ACCEPTABLE	\$x \$y
8	23,762	ACCEPTABLE	\$x \$y
9	66,175	ACCEPTABLE	\$x \$y

Полу-синхронизация



```
class Box {
    int x;
    public Box(int v) {
        x = v;
    }
}

class RacyBoxy {
    Box box;
    public synchronized void set(Box v) {
        box = v;
    }

    public Box get() {
        return box;
    }
}
```



```
@JCStressTest
@State
public class SynchronizedPublish {
    RacyBoxy boxie = new RacyBoxy();

    @Actor
    void actor() {
        boxie.set(new Box(42)); // set is synchronized
    }

    @Actor
    void observer(IntResult1 r) {
        Box t = boxie.get(); // get is not synchronized
        if (t != null) {
            r.r1 = t.x;
        } else {
            r.r1 = -1;
        }
    }
}
```

x86

(fork: #1, iteration #1, JVM args: [-server])

Observed state	Occurrences	Expectation	Interpretation
-1	43,265,036	ACCEPTABLE	Not ready yet
0	0	ACCEPTABLE	Field is not visible yet
42	1,233,714	ACCEPTABLE	Everything is visible

x86

(fork: #1, iteration #1, JVM args: [-server])

Observed state	Occurrences	Expectation	Interpretation
-1	43,265,036	ACCEPTABLE	Not ready yet
0	0	ACCEPTABLE	Field is not visible yet
42	1,233,714	ACCEPTABLE	Everything is visible

POWER-PC

(fork: #1, iteration #1, JVM args: [-server])

Observed state	Occurrences	Expectation	Interpretation
-1	362,286,539	ACCEPTABLE	Not ready yet
0	2341	ACCEPTABLE	Field is not visible yet
42	616,150	ACCEPTABLE	Everything is visible

Вolatile нас (не) спасет!

```
@JCStressTest
@State
public class SynchronizedPublish_VolatileMeh {
    volatile RacyBoxy boxie = new RacyBoxy();

    @Actor
    void actor() {
        boxie.set(new Box(42));
    }

    @Actor
    void observer(IntResult1 r) {
        Box t = boxie.get();
        if (t != null) {
            r.r1 = t.x;
        } else {
            r.r1 = -1;
        }
    }
}
```

Volatile array?

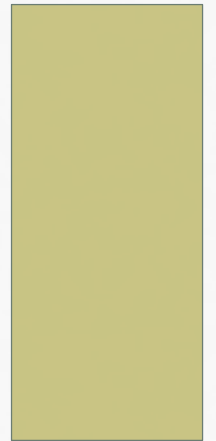
```
@JCStressTest
@State
class VolatileArray {
    volatile int[] arr = new int[2];
    @Actor
    void actor() {
        int[] a = arr;
        a[0] = 1;
        a[1] = 1;
    }
    @Actor
    void observer(IntResult2 r) {
        int[] a = arr;
        r.r1 = a[1];
        r.r2 = a[0];
    }
}
```

Ha **POWER-PC**

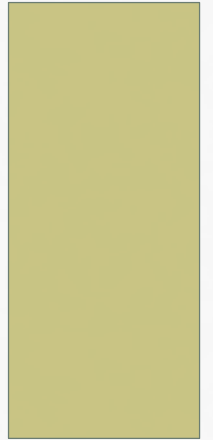
(fork: #1, iteration #1, JVM args: [-server])

Observed state	Occurrences	Expectation	Interpretation
0, 0	704,015	ACCEPTABLE	Everything else is acceptable too.
0, 1	1,291	ACCEPTABLE	Everything else is acceptable too.
1, 0	118	ACCEPTABLE_INTERESTING	Ordering? You wish.
1, 1	37,136,486	ACCEPTABLE	Everything else is acceptable too.

Volatile from (not) experts



A



```
@JCStressTest
@State
public class ReleaseOrderWrong {
    int x;
    volatile int g;
    @Actor
    public void actor1() {
        g = 1;
        x = 1;
    }
    @Actor
    public void actor2(IntResult2 r) {
        r.r1 = g;
        r.r2 = x;
    }
}
```

B



```
public class MyList {  
  
    private volatile List<Integer> list;  
  
    void prepareList() {  
        list = new ArrayList();  
        list.add(1);  
        list.add(2);  
    }  
  
    List<Integer> getMyList() {  
        return list;  
    }  
}
```

```
@JCStressTest
```

```
@State
```

```
public class ReleaseOrder2Wrong {  
    volatile List<Integer> list;  
    @Actor  
    public void actor1() {  
        list = new ArrayList<>();  
        list.add(42);  
    }  
    @Actor  
    public void actor2(IntResult1 r) {  
        List<Integer> l = list;  
        if (l != null) {  
            if (l.isEmpty()) {  
                r.r1 = 0;  
            } else  
                r.r1 = l.get(0);  
        } else  
            r.r1 = -1;  
    }  
}
```

(fork: #1, iteration #1, JVM args: [-server])

Observed state	Occurrences	Expectation	Interpretation
-1	65,119,848	ACCEPTABLE	Reading null list
0	252,169	ACCEPTABLE_INTERESTING	List is not fully populated
42	1,980,313	ACCEPTABLE	Reading a fully populated list



C


```
@JCStressTest
@State
public class AcquireOrderWrong {
    int x;
    volatile int g;

    @Actor
    public void actor1() {
        x = 1;
        g = 1;
    }
    @Actor
    public void actor2(IntResult2 r) {
        r.r1 = x;
        r.r2 = g;
    }
}
```

(fork: #1, iteration #1, JVM args: [-server])

Observed state	Occurrences	Expectation	Interpretation
0, 0	60,839,389	ACCEPTABLE	All other cases are acceptable.
0, 1	579	ACCEPTABLE	All other cases are acceptable.
1, 0	41,053	ACCEPTABLE	All other cases are acceptable.
1, 1	40,122,239	ACCEPTABLE	All other cases are acceptable.

D



```
@JCStressTest
@State
public class AcquireOrderWrong {
    int x;
    volatile int g;

    @Actor
    public void actor1() {
        g = 1;
        x = 1;
    }
    @Actor
    public void actor2(IntResult2 r) {
        r.r1 = x;
        r.r2 = g;
    }
}
```

(fork: #1, iteration #1, JVM args: [-server])

Observed state	Occurrences	Expectation	Interpretation
0, 0	108,771,152	ACCEPTABLE	All other cases are acceptable.
0, 1	1,137,881	ACCEPTABLE	All other cases are acceptable.
1, 0	15,218	ACCEPTABLE	All other cases are acceptable.
1, 1	29,451,719	ACCEPTABLE	All other cases are acceptable.

E



```
@JCStressTest
```

```
@State
```

```
public class SafePublication {
```

```
    int x;
```

```
    volatile int ready;
```

```
@Actor
```

```
public void actor1() {
```

```
    x = 1;
```

```
    ready = 1;
```

```
}
```

```
@Actor
```

```
public void actor2(IntResult2 r) {
```

```
    r.r1 = ready;
```

```
    r.r2 = x;
```

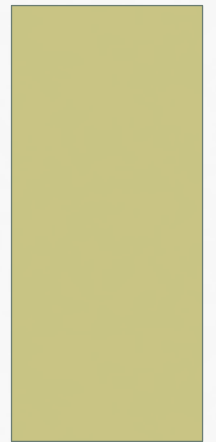
```
}
```

```
}
```

(fork: #1, iteration #1, JVM args: [-server])

Observed state	Occurrences	Expectation	Interpretation
0, 0	69,358,115	ACCEPTABLE	All other cases are acceptable.
0, 1	2,402,453	ACCEPTABLE	All other cases are acceptable.
1, 0	0	FORBIDDEN	Happens-before violation
1, 1	44,989,512	ACCEPTABLE	All other cases are acceptable.

Mirror magic



```
@JCStressTest
@State
public class ReadAfterReadTest {
    int a;

    @Actor
    void actor1() {
        a = 1;
    }

    @Actor
    void actor2(IntResult2 r) {
        r.r1 = a;
        r.r2 = a;
    }
}
```

(fork: #1, iteration #1, JVM args: [-server])

Observed state	Occurrences	Expectation	Interpretation
0, 0	16,736,450	ACCEPTABLE	Doing both reads early.
0, 1	3,941	ACCEPTABLE	Doing first read early, not surprising.
1, 0	84,477	ACCEPTABLE_INTERESTING	First read seen racy value early, and second ... WTF?
1, 1	108,816,262	ACCEPTABLE	Doing both reads late.

Следствие

```
//есть у нас такой класс
public class T {
    public int field1;

    public T( final int field1 ){
        this.field1 = field1;
    }
}

//...где-то в дебрях кода...
public T sharedRef;

//Thread 1
sharedRef = new T( 10 );

//Thread 2
System.out.println( "sharedRef.field1 = " +
sharedRef.field1 );
```

```
//есть у нас такой класс
```

```
public class T {  
    public int field1;  
  
    public T( final int field1 ){  
        this.field1 = field1;  
    }  
}
```

```
//...где-то в дебрях кода...
```

```
public T sharedRef;
```

```
//Thread 1
```

```
sharedRef = new T( 10 );
```

```
//Thread 2
```

```
System.out.println( "sharedRef.field1 = " +  
sharedRef.field1 );
```

NPE!

```
//Thread 2
```

```
if( sharedRef != null ){
```

```
    System.out.println("sharedRef.field1 = "  
                        + sharedRef.field1 );
```

```
} else {
```

```
    //...
```

```
}
```

```
//Thread 2
```

```
if( sharedRef != null ){
```

```
    System.out.println("sharedRef.field1 = "  
                        + sharedRef.field1 );
```

```
} else {
```

```
    //...
```

```
}
```

NPE!


```
//Thread 2
```

```
final T localRef = sharedRef;
```

```
if( localRef != null ){
```

```
    System.out.println("sharedRef.field1 = "  
                        + localRef.field1 );
```

```
} else {
```

```
    //...
```

```
}
```

```
//Thread 2
```

```
final T localRef = sharedRef;
```

```
if( localRef != null ){
```

```
    System.out.println("sharedRef.field1 = "  
                        + localRef.field1 );
```

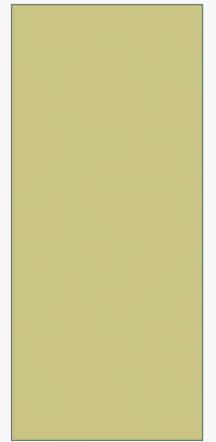
```
} else {
```

```
    //...
```

```
}
```

0!

Safe publication & TSO



```
@JCStressTest
@State
public class UnsafePublication {
    int x = 1;
    MyObject o; // non-volatile, race
    @Actor
    public void publish() {
        o = new MyObject(x);
    }
    @Actor
    public void consume(IntResult1 res) {
        MyObject lo = o;
        if (lo != null) {
            res.r1 = lo.x00 + lo.x01 + lo.x02 + lo.x03;
        } else
            res.r1 = -1;
    }
    static class MyObject {
        int x00, x01, x02, x03;
        public MyObject(int x) {
            x00 = x; x01 = x; x02 = x; x03 = x;
        }
    }
}
```

ДАЖЕ НА x86!

(fork: #1, iteration #1, JVM args: [-server])

Observed state	Occurrences	Expectation	Interpretation
-1	86,515,664	ACCEPTABLE	The object is not yet published
0	751	ACCEPTABLE	Object is published, but all fields are 0.
1	297	ACCEPTABLE	Object is published, at least 1 field is visible.
2	211	ACCEPTABLE	Object is published, at least 2 fields are visible.
3	953	ACCEPTABLE	Object is published, at least 3 fields are visible.
4	4,057,524	ACCEPTABLE	Object is published, all fields are visible.