## Is it possible to do a `reinterpret_cast` in Java?

Well, at first glance someone could say “No, there is no such construction in Java”. And they would be right. That’s because the Object in Java is not just a collection of fields, it contains the information about the object itself — the object *header*. In most JVMs, for 64-bit architectures the header size is 128 bits. Those are a **Mark Word** (information about the state of the object: GC notes, associated lock etc.) and a **Klass Word** (pointer to the information about the class of the object).

### So, there is nothing we could do?

Not exactly.

We’ll need an instance of the class called `com.misc.Unsafe`. It’s a singleton class, and the instance is returned by the method `getUnsafe`

```
public static void main(String[] args) {
    Unsafe unsafe = Unsafe.getUnsafe();
}
```

```
Exception in thread "main" java.lang.SecurityException: Unsafe
  at jdk.unsupported/sun.misc.Unsafe.getUnsafe(Unsafe.java:99)
  at org.ldemetrios.Main.main(Main.java:10)
```

A shame. We aren’t supposed to use it. *Normally.*

OK, let’s bring the heavy artillery in.

```
public static void main(String[] args) throws NoSuchFieldException,
IllegalAccessException {
    Field f = Unsafe.class.getDeclaredField("theUnsafe");
    f.setAccessible(true);
    Unsafe unsafe = (Unsafe) f.get(null);
    System.out.println(unsafe);
}
```

```
sun.misc.Unsafe@659e0bfd
```

Well, that works. Now let’s declare two classes with one field each.

```
static class A {
    long field = 566;
}

static class B {
    long field = 30;
}

public static void main(String[] args) throws NoSuchFieldException,
IllegalAccessException {
    Field f = Unsafe.class.getDeclaredField("theUnsafe");
    f.setAccessible(true);
    Unsafe unsafe = (Unsafe) f.get(null);

    A a = new A();
    System.out.println(unsafe.getLong(a, 0));
    System.out.println(unsafe.getLong(a, 8));
    System.out.println(unsafe.getLong(a, 16));
}
```

```
1
16779776
566
```

Our expectations are satisfied. There are `mark` word, then `klass` word, then the `field`. Let’s open the Hotspot JVM specifications. Last two bits equal to `01` indicate that the object is unlocked. In fact, let’s play with it:

```
    System.out.println(unsafe.getLong(a, 0));
    System.out.println(a);
    System.out.println(unsafe.getLong(a, 0));
    synchronized (a) {
        System.out.println(unsafe.getLong(a, 0));
    }
    System.out.println(unsafe.getLong(a, 0));
```

```
1
org.ldemetrios.Main$A@6d06d69c
468266163201
139861868669160
468266163201
```

Wait, what have just happened? First of all, we asked to print `a`, it called `a.toString()`, which in turn called `a.hashCode()`. Hash code turned out to be `6d06d69c`, and it was cached in the object header. `Mark` word is now `0000006d06d69c01`. The we called `synchronized` on it, the lock was created, then we unlocked it.

The `klass` word, however, never changed. Now let’s change it!

```
    Object a = new A();
    Object b = new B();
    System.out.println(a.getClass());
    unsafe.putLong(a, 8, unsafe.getLong(b, 8));
    System.out.println(a.getClass());
```

```
class org.ldemetrios.Main$A
class org.ldemetrios.Main$B
```

OK, but what happens if we cast it to `B`? Nothing special. Now the `JVM` is completely sure that what lies behind `a` is actually `B`.

```
    B itsSoWrong = (B) a;
    System.out.println(itsSoWrong.field);
```

```
566
```

But don’t forget. Everything is Java is by reference. What if we had a reference of type `A` left?

```
    A a = new A();
    B b = new B();
    unsafe.putLong(a, 8, unsafe.getLong(b, 8));
    B itsSoWrong = (B) a;
    System.out.println(itsSoWrong.field);
```

```
/tmp/escaping-5196877922068559793/src/main/java/org/ldemetrios/Main.java:24:
error: incompatible types: A cannot be converted to B
    B itsSoWrong = (B) a;
                       ^
1 error
```

Oh wait, it’s a compilation error. There is no way `a` can actually point to some `B`, right? *Right?*

```
    A a = new A();
    Object probablyA = a;
    B b = new B();
    unsafe.putLong(a, 8, unsafe.getLong(b, 8));
    B itsSoWrong = (B) probablyA;
    System.out.println(itsSoWrong.field);
    System.out.println(a.getClass());
    System.out.println(a.field);
```

```
566
class org.ldemetrios.Main$B
566
```

Surprisinlgy still seems to be fine.

```
    A thatWasA = (A) (Object) a;
```

```
Exception in thread "main" java.lang.ClassCastException: class
org.ldemetrios.Main$B cannot be cast to class org.ldemetrios.Main$A
(org.ldemetrios.Main$B and org.ldemetrios.Main$A are in unnamed module of loader
'app')
  at org.ldemetrios.Main.main(Main.java:26)
```

Yeah, that broke. `a` is not `A` anymore.