

# Safe Initialization

Final Keyword & Java Memory Model

# **final Keyword**

**final can be used with class, variable, method**

- (1) Restrict changing value of variable**
- (2) Restrict method overriding**
- (3) Restrict inheritance**

# Final Keyword

- Create unmodifiable references
- Restrict method overriding
- Restrict class inheritance
- Part of Java Memory Model
- Guarantees a field visibility in a multi-threaded application
- Safe initialization for objects, arrays and collections

```
public class NonFinalVariable {  
    public Integer value;  
  
    public NonFinalVariable() {  
        this.value = 1;  
    }  
}
```

```
public class FinalVariable {  
    public final Integer value;  
  
    public FinalVariable() {  
        this.value = 1;  
    }  
}
```

```
public class FinalVariable {  
  
    public final int[] value;  
  
    public FinalVariable() {  
        this.value = new int[5];  
        this.value[0] = 1;  
        this.value[1] = 2;  
        this.value[2] = 3;  
    }  
}
```

```
public class FinalVariable {  
  
    public final List<Integer> value;  
  
    public FinalVariable() {  
        this.value = new ArrayList<>();  
        this.value.add(1);  
        this.value.add(2);  
        this.value.add(3);  
    }  
}
```

```
public class FinalVariable {  
    public final InnerClass value;  
  
    public FinalVariable() {  
        this.value = new InnerClass( val: 1);  
    }  
  
    private static class InnerClass {  
        public int innerValue;  
  
        public InnerClass(int val) {  
            this.innerValue = val;  
        }  
    }  
}
```

```
public class PartiallyFinalVariables {  
  
    public final int value1;  
    public int value2;  
    public int value3;  
  
    public PartiallyFinalVariables() {  
        this.value1 = 1;  
        this.value2 = 2;  
        this.value3 = 3;  
    }  
}
```

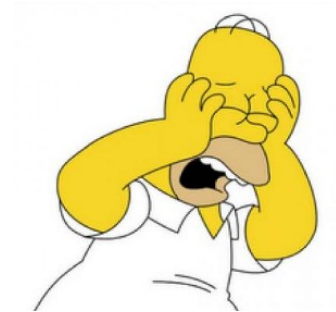
```
public class FinalVariables {  
  
    public final int value1;  
    public final int value2;  
    public final int value3;  
  
    public FinalVariables() {  
        this.value1 = 1;  
        this.value2 = 2;  
        this.value3 = 3;  
    }  
}
```

**<< jcstress demo >>**

<http://openjdk.java.net/projects/code-tools/jcstress/>



# !! I can't use final !!



- Synchronized block (implicit locks)
- Explicit locks, volatile keyword
- CAS operations – Atomic\*, \*Adder
- java.util.concurrent package
  - SynchronizedMap, ConcurrentHashMap
  - CopyOnWriteArray(List|Set),  
Synchronized(List|Set)
  - BlockingQueue, ConcurrentLinkedQueue
- VarHandles, ? Unsafe ?

# Summary

Always start with a FINAL variable and change it only if the object's state is really supposed to be mutable.

.. then you need to handle visibility problems using memory barriers generated by LOCKS or VOLATILE.