Introduction to Safe Publication in Java

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

- Understand what “safe publication” means in the context of Java objects running in concurrent programs
Overview of Safe Publication in Java
Overview of Safe Publication in Java

- A Java object that’s shared across threads must meet several criteria

See flylib.com/books/en/2.558.1/safe_publication.html
Overview of Safe Publication in Java

- A Java object that’s shared across threads must meet several criteria
- They must be constructed properly

See shipilev.net/blog/2014/safe-public-construction
Overview of Safe Publication in Java

- A Java object that’s shared across threads must meet several criteria
- They must be constructed properly
- If the this reference “escapes” during construction the object is *not* properly constructed

public class ThisEscape {
    public ThisEscape (EventSource source) {
        source .registerListener
            (new EventListener() {
                public void onEvent( Event event) {
                    doSomething(event);
                }
            });
    }
}

See vlkan.com/blog/post/2014/02/14/java-safe-publication
Overview of Safe Publication in Java

- A Java object that’s shared across threads must meet several criteria
  - They must be constructed properly
  - If the `this` reference “escapes” during construction the object is *not* properly constructed

```java
public class ThisEscape {
    public ThisEscape (EventSource source) {
        source .registerListener
            (new EventListener() {
                public void onEvent (Event event){
                    doSomething(event);
                }
            });
    }
}
```

*Implicitly publishes the enclosing ThisEscape object because inner class instances contain a hidden reference to the enclosing object*

See vlkan.com/blog/post/2014/02/14/java-safe-publication
Overview of Safe Publication in Java

• A Java object that’s shared across threads must meet several criteria
  • They must be constructed properly
  • They must be “published safely”

See shipilev.net/blog/2014/safe-public-construction
Overview of Safe Publication in Java

- A Java object that’s shared across threads must meet several criteria
  - They must be constructed properly
  - They must be “published safely”
    - Safe publication ensures all values written within a thread *before* the publication are visible to all reader threads that observe the published object

This “object-level” property can be viewed as a generalization of “operation-level” atomic operations discussed in earlier lessons
A Java object that’s shared across threads must meet several criteria:

- They must be constructed properly
- They must be “published safely”
  - Safe publication ensures all values written within a thread before the publication are visible to all reader threads that observe the published object
- Storing an object reference into a public field is insufficient to publish that object safely

```java
class A {
    public ArrayList<String> mList;
    public void initialize() {
        mList = new ArrayList<>(Array.asList(...));
    }
}

// Thread T1
A a = new A();
a.initialize();

// Thread T2
doSomething(a.mList);
```

This problem only arises in multi-threaded programs on multi-core CPUs
A Java object that’s shared across threads must meet several criteria:

- They must be constructed properly.
- They must be “published safely.”
  - Safe publication ensures all values written within a thread before the publication are visible to all reader threads that observe the published object.
- Storing a object reference into a public field is insufficient to publish that object safely.

```java
class A {
    public ArrayList<String> mList;

    public void initialize() {
        mList = new ArrayList<>(Array.asList(...));
    }
}
```

```
// Thread T1
A a = new A();
a.initialize();
```

```
// Thread T2
doSomething(a.mList);
```
Overview of Safe Publication in Java

- A Java object that’s shared across threads must meet several criteria
  - They must be constructed properly
  - They must be “published safely”
    - Safe publication ensures all values written within a thread *before* the publication are visible to all reader threads that observe the published object
  - Storing a object reference into a public field is insufficient to publish that object safely

```java
class A {
    public ArrayList<String> mList;

    public void initialize() {
        mList = new ArrayList<>((
            Array.asList(...);
        }
    }

    // Thread T1
    A a = new A();
    a.initialize();

    // Thread T2
    doSomething(a.mList);
}
```

*mList is not guaranteed to be initialized when thread *T2* gets a reference to object *a*
End of Introduction to Safe Publication in Java