Applying Java Structured Concurrency: Case Study ex5

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

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

• Understand Java’s structured concurrency model
• Recognize the classes used to program Java’s structure concurrency model
• Evaluate the design & performance of various Java concurrency models
• Know how to implement a custom StructuredTaskScope
• Case study ex5 shows how to create & apply a custom StructuredTaskScope

public class ShutdownOnNonNullSuccess<T> extends StructuredTaskScope<T> {
    private volatile T mResult;
    protected void handleComplete (Future<T> future) {
        ... 
        T result = future.resultNow();
        if (result != null) {
            mResult = result; shutdown();
        }
    }

    public T result()
    { return mResult; } ...
}
Applying Reactive Java Concurrency to Case Study ex5
Applying Reactive Java Concurrency to Case Study ex5

```java
/**
 * A `{@link StructuredTaskScope}` that captures the result of the first
 * subtask to complete successfully (i.e., without returning `{@code
 * null}`) or returns `{@code null}` if no subtask completes
 * successfully. Once captured, it invokes the `{@code shutdown()}`
 * method to interrupt unfinished threads and wakeup the owner.
 *
 * The policy implemented by this class is intended for cases where
 * the result of any subtask will do ("invoke any") and where the
 * results of other unfinished subtask are no longer needed.
 */

public class ShutdownOnNonNullSuccess<T> extends StructuredTaskScope<T> {
    private volatile T mResult;

    /**
     * Creates an unnamed structured task scope that creates virtual
     * threads.
     */
    public ShutdownOnNonNullSuccess() { super(new NullThreadFactory());
```
End of Applying Java Structured Concurrency: Case Study ex5