The LockManager App Case Study: Server Structure & Functionality (Part 2)

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

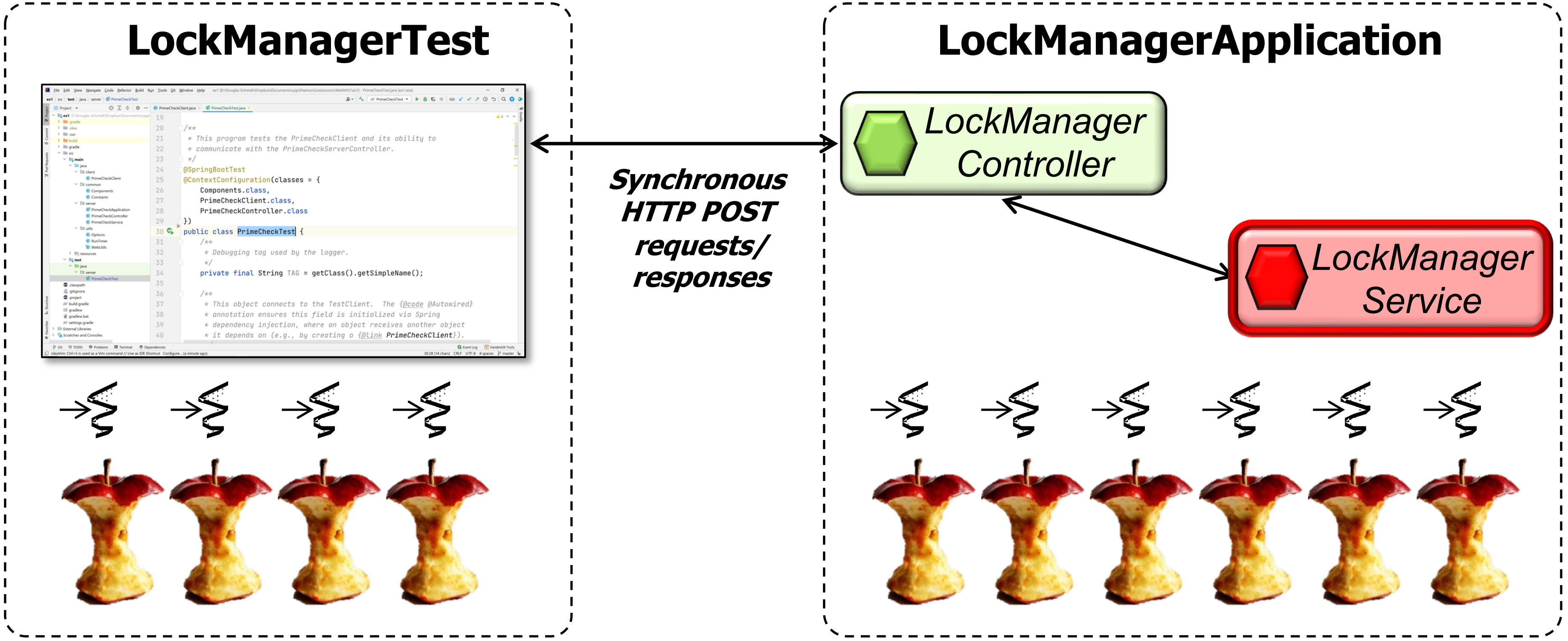
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- This lesson shows the structure & functionality of the service class for the LockManager microservice developed using Spring WebMVC



```
/**
 * This program tests the PrimeCheckClient and its ability to
 * communicate with the PrimeCheckServerController.
 */
@SpringBootTest
@ContextConfiguration(classes = {
    Components.class,
    PrimeCheckClient.class,
    PrimeCheckController.class
})
public class PrimeCheckTest {
    /**
     * Debugging tag used by the logger.
     */
    private final String TAG = getClass().getSimpleName();

    /**
     * This object connects to the TestClient.  The {@code @Autowired}
     * annotation ensures this field is initialized via Spring
     * dependency injection, where an object receives another object
     * it depends on (e.g., by creating a {@link PrimeCheckClient}).
```

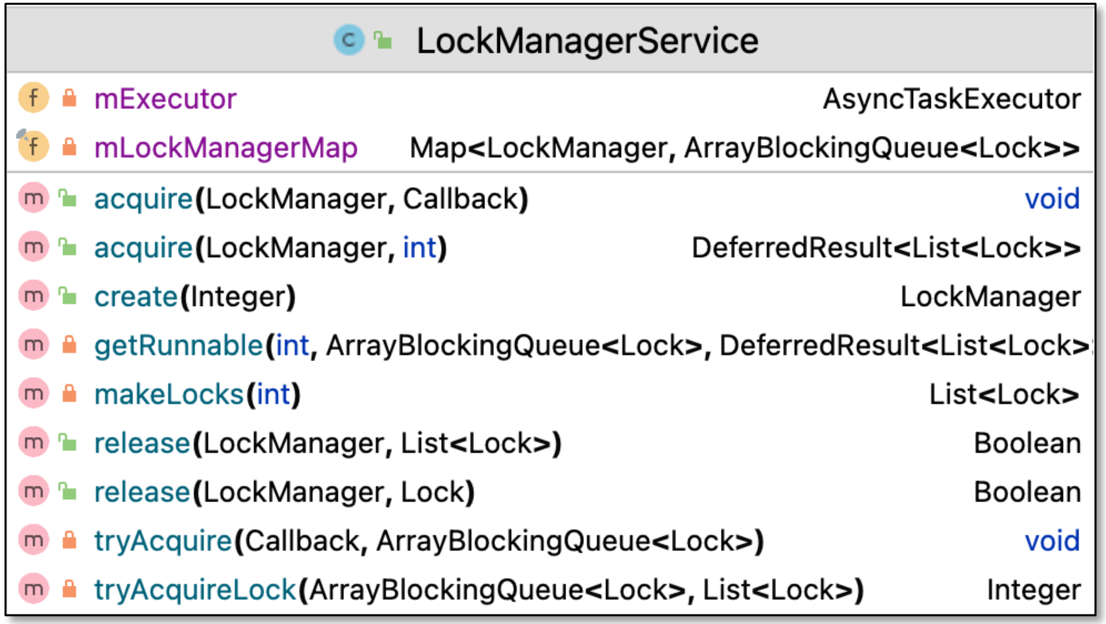
See WebMVC/ex5/src/main/java/edu/vandy/lockmanager/server

Structure & Functionality of the LockManagerService

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
```



LockManagerService

mExecutor	AsyncTaskExecutor
mLockManagerMap	Map<LockManager, ArrayBlockingQueue<Lock>>
acquire(LockManager, Callback)	void
acquire(LockManager, int)	DeferredResult<List<Lock>>
create(Integer)	LockManager
getRunnable(int, ArrayBlockingQueue<Lock>, DeferredResult<List<Lock>>	
makeLocks(int)	List<Lock>
release(LockManager, List<Lock>)	Boolean
release(LockManager, Lock)	Boolean
tryAcquire(Callback, ArrayBlockingQueue<Lock>)	void
tryAcquireLock(ArrayBlockingQueue<Lock>, List<Lock>)	Integer

See WebMVC/ex5/src/main/java/edu/vandy/lockmanager/server/LockManagerService.java

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
```

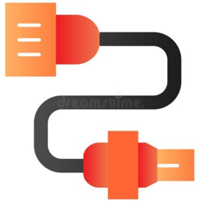
This annotation indicates the class implements "business logic" & enables auto-detection & wiring of dependent classes via classpath scanning

See www.baeldung.com/spring-component-repository-service

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  @Autowired
  private AsyncTaskExecutor mExecutor;
```



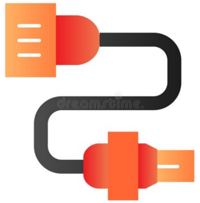
This auto-wired field uses Executors.newVirtualThreadPerTaskExecutor() to run deferred HTTP request processing off the receiving servlet thread

See davidvlijmincx.com/posts/create_virtual_threads_with_project_loom

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  @Autowired
  private AsyncTaskExecutor mExecutor;
```



This mExecutor field is autowired to a @Bean that creates a virtual thread per task

```
@Bean(APPLICATION_TASK_EXECUTOR_BEAN_NAME)
public AsyncTaskExecutor asyncTaskExecutor() {
    return new TaskExecutorAdapter
      (Executors.newVirtualThreadPerTaskExecutor());
}
```

See WebMVC/ex5/src/main/java/edu/vandy/lockmanager/common/ServerBeans.java

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  @Autowired
  private AsyncTaskExecutor mExecutor;



  @Autowired
  private Map<LockManager, ArrayBlockingQueue<Lock>>
    mLockManagerMap;
```

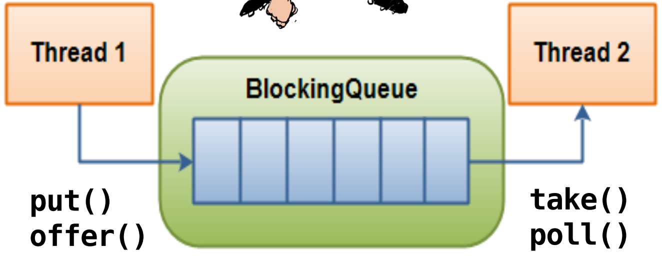
This autowired field associates a LockManager object with its ArrayBlockingQueue state info

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  @Autowired
  private AsyncTaskExecutor mExecutor;
```

Limits concurrent access to the fixed number of available locks associated with the LockManager



```
  @Autowired
  private Map<LockManager, ArrayBlockingQueue<Lock>>
    mLockManagerMap;
```

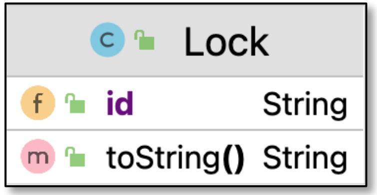
See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ArrayBlockingQueue.html

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  @Autowired
  private AsyncTaskExecutor mExecutor;

  @Autowired
  private Map<LockManager, ArrayBlockingQueue<Lock>>
    mLockManagerMap;
```



Define an object that clients can use as a lock in a distributed system

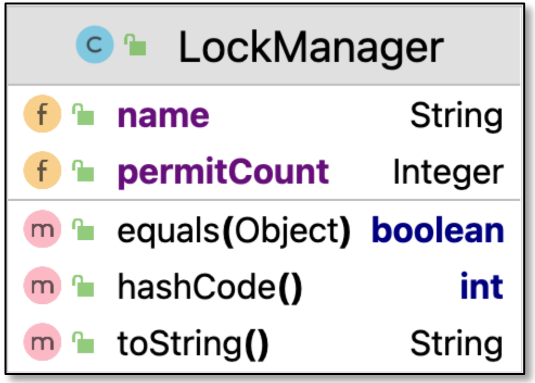
Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  @Autowired
  private AsyncTaskExecutor mExecutor;
```

Used in conjunction with a Map to keep track of allocated ArrayBlockingQueue objects

```
  @Autowired
  private Map<LockManager, ArrayBlockingQueue<Lock>>
    mLockManagerMap;
```



LockManager

name	String
permitCount	Integer
equals(Object)	boolean
hashCode()	int
toString()	String

See WebMVC/ex5/src/main/java/edu/vandy/lockmanager/common/LockManager.java

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  @Autowired
  private AsyncTaskExecutor mExecutor;
```

A Map is used to associate LockManager objects with ArrayBlockingQueue objects

```
  @Autowired
  private Map<LockManager, ArrayBlockingQueue<Lock>>
    mLockManagerMap;
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.html

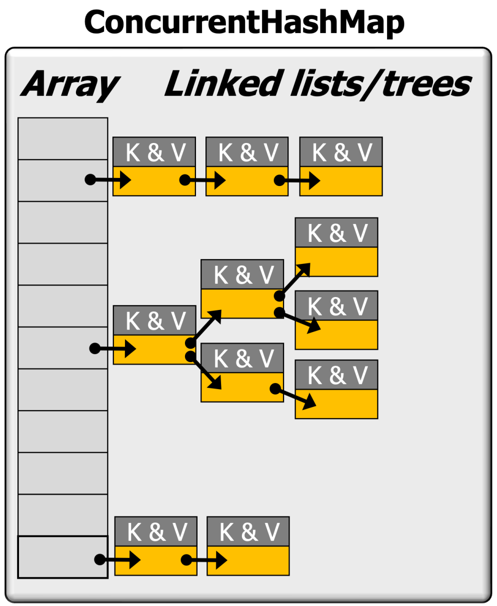
Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  @Autowired
  private AsyncTaskExecutor mExecutor;
```

This Map is autowired to a ConcurrentHashMap

```
  @Autowired
  private Map<LockManager, ArrayBlockingQueue<Lock>>
    mLockManagerMap;
```



See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.html

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  ...
  public LockManager create(Integer permits) {...}
```

This factory method returns a LockManager that implements a distributed semaphore

```
  @Async public void acquire(LockManager lockManager,
                             Callback callback) {...}


  public DeferredResult<List<Lock>>
    acquire(LockManager lockManager, Integer permits) {.}
  ...
```

Each instance of LockManager handles a different set of permits/locks

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  ...
  public LockManager create(Integer permits) {...}
```

This method uses a Java ArrayBlockingQueue to acquire a single permit/lock

```
  @Async public void acquire(LockManager lockManager,
                             Callback callback) {...}


  public DeferredResult<List<Lock>>
    acquire(LockManager lockManager, Integer permits) {.}
  ...
```

See upcoming part of the lessons for implementation details

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
   ...
   public LockManager create(Integer permits) {...}

   @Async public void acquire(LockManager lockManager,
                              Callback callback) {...}


   public DeferredResult<List<Lock>>
     acquire(LockManager lockManager, Integer permits) {.}
   ...
```

This annotation marks acquire() as automatically being executed asynchronously by the AsyncTaskExecutor in a virtual thread

See org/springframework/scheduling/annotation/Async.html

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  ...
  public LockManager create(Integer permits) {...}




  @Async public void acquire(LockManager lockManager,
                             Callback callback) {...}


  public DeferredResult<List<Lock>>
    acquire(LockManager lockManager, Integer permits) {.}
```

This method uses a Java ArrayBlockingQueue to acquire multiple permits/locks

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  ...
  public LockManager create(Integer permits) {...}


  @Async public void acquire(LockManager lockManager,
                             Callback callback) {...}


  public DeferredResult<List<Lock>>
    acquire(LockManager lockManager, Integer permits) {.}
  ...
```

Computation is manually offloaded from HTTP worker thread to a virtual thread

See www.baeldung.com/spring-deferred-result

Structure & Functionality of the LockManagerService

- LockManagerService defines methods called by LockManagerController, which implements a distributed semaphore using a Java ArrayBlockingQueue

```
@Service
public class LockManagerService {
  ...
  public Boolean release(LockManager lockManager,
                         Lock lock) {...}
  public Boolean release(LockManager lockManager,
                         List<Lock> locks) {...}
  ...
```

These methods use the Java ArrayBlockingQueue to return one or more permits/locks to the semaphore

See upcoming part of the lessons for implementation details

End of the LockManager App Case Study: Server Structure & Functionality (Part 2)