#	Source Code		Simplified Code	
	Original	Modified	Original	Modified

1	`LeafQueue leafQueue = ...;`
	`-synchronized (leafQueue) {`
	
	`  57 LOC`
	
	`}`
		`LeafQueue leafQueue = ...;`
		`+try {`
		`+  leafQueue.getReadLock().lock();`
		`  57 LOC`
		`+} finally {`
		`+  leafQueue.getReadLock().unlock();`
		`}`
			`synchronized (obj)`
			`{`
			`  ...`
			`}`
				`try {`
				`  obj.lock();`
				`  ...`
				`} finally {`
				`  obj.unlock();`
				`}`

2	`-Lock readlock =`
	`-  classLoaderContainerMapLock.readLock();`
	`-try {`
	`-  readlock.lock();`
	`-  result = classLoaderContainerMap.get(tccl);`
	`-} finally {`
	`-  readlock.unlock();`
	`-}`
	`-if (result == null) {`
	`-  Lock writelock =`
	`-  classLoaderContainerMapLock.writeLock();`
	`-  try {`
	`-  writeLock.lock();`
	`  result = classLoaderContainerMap.get(tccl);`
	`  if (result == null) {`
	`  result = new ServerContainerImpl();`
	`  classLoaderContainerMap.put(tccl,result);`
	`  }`
	`-  } finally {`
	`-  writeLock.unlock();`
	`-  }`
	`}`
		`+synchronized (classLoaderContainerMapLock) {`
		
		`  result = classLoaderContainerMap.get(tccl);`
		`  if (result == null) {`
		`  result = new ServerContainerImpl();`
		`  classLoaderContainerMap.put(tccl,result);`
		`  }`
		
		`}`
			`try {`
			`  readLock.lock();`
			`  read operations`
			`} finally {`
			`  readLock.unlock();`
			`}`
			`try {`
			`  writeLock.lock();`
			`  write operations`
			`} finally {`
			
			`writeLock.unlock();`
			`}`
				`synchronized {`
				`  all operations`
				`}`

3	`private static final Object lock = new`
	`  Object();`
	`private Map<...> count = new HashMap<>();`
	`-synchronized (count) {`
	`-  Pair<Job, String> key =`
	`-  new ImmutablePair<>(jobID, name);`
	
	`-  if (count.containsKey(key)) {`
	`-  count.put(key, count.get(key) + 1);`
	`  } else {`
	`-  count.put(key, 1);`
	`  }`
	`}`
		`private static final Object lock = new`
		`  Object();`
		`private Map<...> count = new HashMap<>();`
		`+synchronized(lock)`
		`+  if (!jobCounts.containsKey(jobID)) {`
		`+  jobCounts.put(jobID, new HashMap<>());`
		`+  }`
		`+  Map<String, Integer> count =`
		`+  jobCounts.get(jobID);`
		`+  if (count.containsKey(name)) {`
		`+  count.put(name, count.get(name) + 1);`
		`  } else {`
		`+  count.put(name, 1);`
		`  }`
		`}`
			`synchronized (obj1)`
			`{`
			`  ...`
			`}`
				`synchronized`
				`(obj2) {`
				`  ...`
				`}`

4	`-public boolean isAccessed() {`
	`  return this.accessed;`
	`}`
		`+public synchronized boolean isAccessed() {`
		`  return this.accessed;`
		`}`
			`void foo() {`
			`  ...`
			`}`
				`synchronized void`
				`foo() {`
				`  ...`
				`}`

5	`-synchronized (this.channelLookup) {`
	`-  try{`
	`  lookupResponse = AkkaUtils.`
	`<JobManagerMessages.ConnectionInformation>ask(ch`
	`annelLookup, new`
	`JobManagerMessages.LookupConnectionInformation(c`
	`onnectionInfo, jobID, sourceChannelID),`
	`timeout).response();`
	`-  }catch(IOException ioe) {`
	`-  throw ioe;`
	`-  }`
	`-}`
		`lookupResponse = AkkaUtils.`
		`<JobManagerMessages.ConnectionInformation>ask(`
		`channelLookup, new`
		`JobManagerMessages.LookupConnectionInformation`
		`(  connectionInfo, jobID, sourceChannelID),`
		`timeout).response();`
			`synchronized (obj)`
			`{`
			`  ...`
			`}`
				`...`

6	`synchronized (buffers) {`
	`  if (...) {`
	`-  if (spillWriter != null) {`
	`-  spillWriter.close();`
	`-  }`
	`  isFinished = true;`
	`  }`
	`}`
		`synchronized (buffers) {`
		`  if (...) {`
		`  isFinished = true;`
		`  }`
		`}`
		`+if (spillWriter != null) {`
		`+  spillWriter.close();`
		`+}`
			`synchronized (obj)`
			`{`
			`  statements1`
			`  statements2`
			`}`
				`synchronized (obj)`
				`{`
				`  statements2`
				`}`
				`statements1`

7	`-public synchronized void reset() {`
	
	`  map.clear();`
	`  members = EMPTY_MEMBERS;`
	
	`}`
		`+private final Object membersLock = new`
		`+  Object();`
		`+public void reset() {`
		`+  synchronized (membersLock) {`
		`  map.clear();`
		`  members = EMPTY_MEMBERS;`
		`+  }`
		`}`
			`synchronized void`
			`foo() {`
			`  ...`
			`}`
				`void foo() {`
				`  synchronized`
				`(obj) {`
				`  ...`
				`  }`
				`}`