Java Synchronized Collections: Example Application

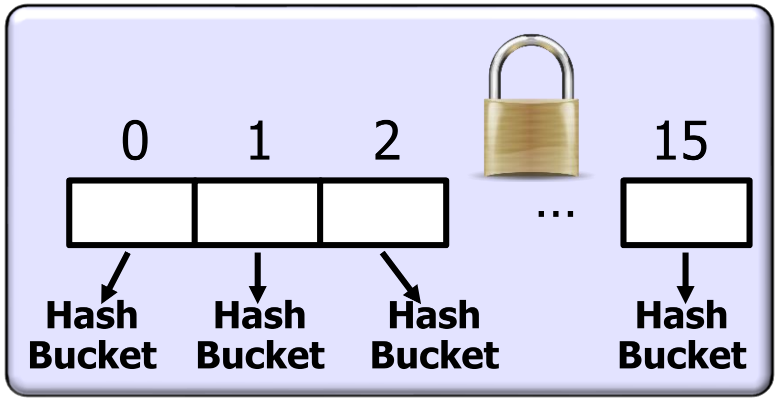
Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Lesson

- Recognize the capabilities & limits of Java's synchronized collections
- Know how to apply Java synchronized map in practice

SynchronizedMap



Applying a Java SynchronizedMap in Practice

Applying a Java SynchronizedMap in Practice

- Apply a Java SynchronizedMap to compute/cache/retrieve prime #'s

```
void runTest(int maxIters,
             Map<Integer, Integer> primeCache) {
  ...
  Random ran = new Random();

  Runnable primeChecker = () -> {
    for (int i = 0; i < maxIters; i++) {
      int primeCandidate = Math.abs(ran.nextInt(maxIters) + 1);


      int smallestFactor =
        primeCache.computeIfAbsent(primeCandidate,
                                   this::isPrime);

        if (smallestFactor != 0) ...
        else ...
    }};
  ...
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex9

Applying a Java SynchronizedMap in Practice

- Apply a Java SynchronizedMap to compute/cache/retrieve prime #'s

```
void runTest(int maxIters,
             Map<Integer, Integer> primeCache) {
  ...
  Random ran = new Random();

  Runnable primeChecker = () -> {
    for (int i = 0; i < maxIters; i++) {
      int primeCandidate = Math.abs(ran.nextInt(maxIters) + 1);


      int smallestFactor =
        primeCache.computeIfAbsent(primeCandidate,
                                   this::isPrime);

        if (smallestFactor != 0) ...
        else ...
    }};
  ...
```

Pass a Map used as a cache

Applying a Java SynchronizedMap in Practice

- Apply a Java SynchronizedMap to compute/cache/retrieve prime #'s

```
void runTest(int maxIters,
             Map<Integer, Integer> primeCache) {
  ...
  Random ran = new Random();

  Runnable primeChecker = () -> {
    for (int i = 0; i < maxIters; i++) {
      int primeCandidate = Math.abs(ran.nextInt(maxIters) + 1);


      int smallestFactor =
        primeCache.computeIfAbsent(primeCandidate,
                                   this::isPrime);

        if (smallestFactor != 0) ...
        else ...
    }};
  ...
```

Create a random # generator

Applying a Java SynchronizedMap in Practice

- Apply a Java SynchronizedMap to compute/cache/retrieve prime #'s

```
void runTest(int maxIters,
             Map<Integer, Integer> primeCache) {
  ...
  Random ran = new Random();

  Runnable primeChecker = () -> {
    for (int i = 0; i < maxIters; i++) {
      int primeCandidate = Math.abs(ran.nextInt(maxIters) + 1);


      int smallestFactor =
        primeCache.computeIfAbsent(primeCandidate,
                                   this::isPrime);

        if (smallestFactor != 0) ...
        else ...
  }};
  ...
```

A lambda runnable that checks if maxIters random #'s are prime

Applying a Java SynchronizedMap in Practice

- Apply a Java SynchronizedMap to compute/cache/retrieve prime #'s

```
void runTest(int maxIters,
             Map<Integer, Integer> primeCache) {
  ...
  Random ran = new Random();

  Runnable primeChecker = () -> {
    for (int i = 0; i < maxIters; i++) {
      int primeCandidate = Math.abs(ran.nextInt(maxIters) + 1);


      int smallestFactor =
        primeCache.computeIfAbsent(primeCandidate,
                                   this::isPrime);

        if (smallestFactor != 0) ...
        else ...
    }};
  ...
```

Get a positive random #

Applying a Java SynchronizedMap in Practice

- Apply a Java SynchronizedMap to compute/cache/retrieve prime #'s

```
void runTest(int maxIters,
             Map<Integer, Integer> primeCache) {
  ...
  Random ran = new Random();

  Runnable primeChecker = () -> {
    for (int i = 0; i < maxIters; i++) {
      int primeCandidate = Math.abs(ran.nextInt(maxIters) + 1);


      int smallestFactor =
        primeCache.computeIfAbsent(primeCandidate,
                                   this::isPrime);

        if (smallestFactor != 0) ...
        else ...
    }};
  ...
```

Check if the factor for this # is already in the cache & if not atomically check if this # is prime & put in the cache

Applying a Java SynchronizedMap in Practice

- Apply a Java SynchronizedMap to compute/cache/retrieve prime #'s

```
void runTest(int maxIters,
             Map<Integer, Integer> primeCache) {
  ...
  Random ran = new Random();

  Runnable primeChecker = () -> {
    for (int i = 0; i < maxIters; i++) {
      int primeCandidate = Math.abs(ran.nextInt(maxIters) + 1);


      int smallestFactor =
        primeCache.computeIfAbsent(primeCandidate,
                                   this::isPrime);

        if (smallestFactor != 0) ...
        else ...
    }};
  ...
```

Handle the results of the prime check computation

Applying a Java SynchronizedMap in Practice

- Apply a Java SynchronizedMap to compute/cache/retrieve prime #'s

```
void runTest(int maxIters,
             Map<Integer, Integer> primeCache)  {
  ...
  Runnable primeChecker = () -> { ...
  };


  for (int i = 0; i < sNUMBER_OF_CORES; i++)
    mExecutor.execute(primeChecker);
  ...
}

static public void main(String[] argv) {
  ...
  runTest(maxIterations,
          Collections.synchronizedMap(new HashMap<>));
  ...
}
```

Create a group of tasks that run the prime checker lambda

Applying a Java SynchronizedMap in Practice

- Apply a Java SynchronizedMap to compute/cache/retrieve prime #'s

```
void runTest(int maxIters,
             Map<Integer, Integer> primeCache)  {
  ...
  Runnable primeChecker = () -> { ...
  };


  for (int i = 0; i < sNUMBER_OF_CORES; i++)
    mExecutor.execute(primeChecker);
  ...
}

static public void main(String[] argv) {
  ...
  runTest(maxIterations,
          Collections.synchronizedMap(new HashMap<>));
  ...
}
```

Run the test function with a synchronized HashMap

See docs.oracle.com/javase/8/docs/api/java/util/Collections.html#synchronizedMap

End of Java Synchronized Collections: Example Application