Java Phaser: 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 Part of the Lesson

- Understand the structure & functionality of the Java Phaser barrier synchronizer
- Recognize the key methods in the Java Phaser
- Know how to program with Java Phaser in practice

```
void runTasks(List<MyTask> tasks) {
  Phaser phaser = new Phaser(1);

  tasks.forEach(task -> {
                phaser.register();

                new Thread(() -> {
                           phaser.arriveAndAwaitAdvance();
                           task.run();
                           }).start();
                });

   phaser.arriveAndDeregister();
 }
```

This program is based on examples in the Java documentation available at docs.oracle.com/javase/8/docs/api/java/util/concurrent/Phaser.html

Test Driver Program Walkthrough

Test Driver Program Walkthrough

- Main entry point into the test program

```
static void main(String[] argv) {

   List<MyTask> tasks = IntStream

      .rangeClosed(1, sNUMBER_OF_TASKS)

      .mapToObj(MyTask::new)

      .collect(toList());


   runTasks(tasks);


   startTasks(tasks, sITERATIONS);
}
```

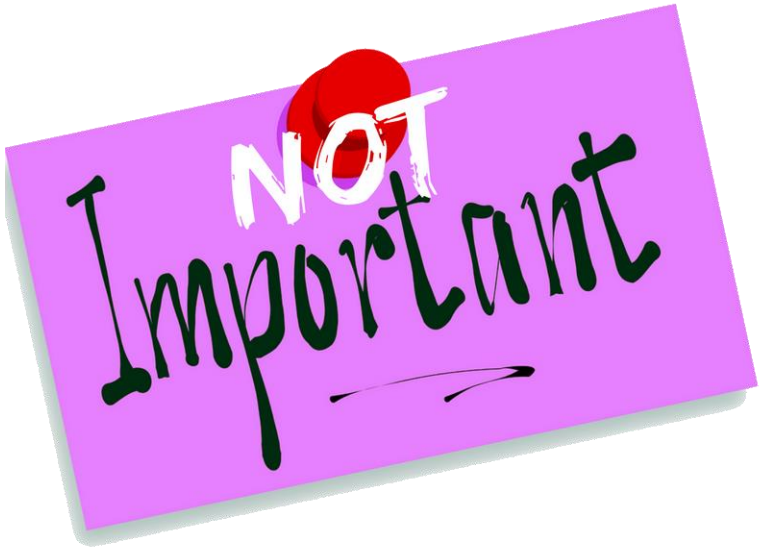
See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex26

Test Driver Program Walkthrough

- Main entry point into the test program

```
static void main(String[] argv) {

    List<MyTask> tasks = IntStream

      .rangeClosed(1, sNUMBER_OF_TASKS)

      .mapToObj(MyTask::new)

      .collect(toList());


    runTasks(tasks);


    startTasks(tasks, sITERATIONS);
}
```

Create a list of MyTask objects



However, the details of what MyTask does are not important for our discussion

Test Driver Program Walkthrough

- Main entry point into the test program

```
static void main(String[] argv) {

    List<MyTask> tasks = IntStream

      .rangeClosed(1, sNUMBER_OF_TASKS)

      .mapToObj(MyTask::new)

      .collect(toList());


    runTasks(tasks);


    startTasks(tasks, sITERATIONS);
}
```

Create a stream from 1 to sNUMBER_OF_TASKS

Test Driver Program Walkthrough

- Main entry point into the test program

```
static void main(String[] argv) {

    List<MyTask> tasks = IntStream

      .rangeClosed(1, sNUMBER_OF_TASKS)

      .mapToObj(MyTask::new)

      .collect(toList());


    runTasks(tasks);


    startTasks(tasks, sITERATIONS);
}
```

Create a new MyTask object for each number in the stream

Test Driver Program Walkthrough

- Main entry point into the test program

```
static void main(String[] argv) {

    List<MyTask> tasks = IntStream

      .rangeClosed(1, sNUMBER_OF_TASKS)

      .mapToObj(MyTask::new)

      .collect(toList());

    runTasks(tasks);


    startTasks(tasks, sITERATIONS);
}
```

Convert the stream into a list of MyTask objects

Test Driver Program Walkthrough

- Main entry point into the test program

```
static void main(String[] argv) {

   List<MyTask> tasks = IntStream

      .rangeClosed(1, sNUMBER_OF_TASKS)

      .mapToObj(MyTask::new)

      .collect(toList());


   runTasks(tasks);


   startTasks(tasks, sITERATIONS);
}
```

Run the test showcasing a one-shot Phaser that runs a list of tasks that all start at the same time

This method illustrates the use of a Phaser as an "entry barrier"

Test Driver Program Walkthrough

- Main entry point into the test program

```
static void main(String[] argv) {

    List<MyTask> tasks = IntStream

      .rangeClosed(1, sNUMBER_OF_TASKS)

      .mapToObj(MyTask::new)

      .collect(toList());


    runTasks(tasks);


    startTasks(tasks, sITERATIONS);
}
```

Run the test that showcases a cyclic Phaser that repeatedly performs actions for a given # of iterations

This method illustrates the use of a Phaser as a "cyclic exit barrier"

Applying a One-shot Phaser with Java

Applying a One-shot Phaser with Java

- A one-shot Phaser that starts running a list of tasks simultaneously

```
void runTasks(List<MyTask> tasks) {
  Phaser phaser = new Phaser(1);

  tasks.forEach(task -> {
                phaser.register();

                new Thread(() -> {
                           phaser.arriveAndAwaitAdvance();
                           task.run();
                           }).start();
                });

   phaser.arriveAndDeregister();
 }
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex26

Applying a One-shot Phaser with Java

- A one-shot Phaser that starts running a list of tasks simultaneously

```
void runTasks(List<MyTask> tasks) {
  Phaser phaser = new Phaser(1);

  tasks.forEach(task -> {
                phaser.register();

                new Thread(() -> {
                           phaser.arriveAndAwaitAdvance();
                           task.run();
                           }).start();
                });

   phaser.arriveAndDeregister();
 }
```

Create a new phaser with a "parties" value of 1 to implicitly register itself

Applying a One-shot Phaser with Java

- A one-shot Phaser that starts running a list of tasks simultaneously

```
void runTasks(List<MyTask> tasks) {
  Phaser phaser = new Phaser(1);

  tasks.forEach(task -> {
                  phaser.register();

                  new Thread(() -> {
                              phaser.arriveAndAwaitAdvance();
                              task.run();
                              }).start();
                  });

   phaser.arriveAndDeregister();
 }
```

Iterate thru all the tasks

Applying a One-shot Phaser with Java

- A one-shot Phaser that starts running a list of tasks simultaneously

```
void runTasks(List<MyTask> tasks) {
  Phaser phaser = new Phaser(1);

  tasks.forEach(task -> {
                phaser.register();

                new Thread(() -> {
                           phaser.arriveAndAwaitAdvance();
                           task.run();
                           }).start();
                });

   phaser.arriveAndDeregister();
 }
```

Inform phaser there's a new party to add

Applying a One-shot Phaser with Java

- A one-shot Phaser that starts running a list of tasks simultaneously

```
void runTasks(List<MyTask> tasks) {
  Phaser phaser = new Phaser(1);

  tasks.forEach(task -> {
                phaser.register();

                new Thread(() -> {
                                  phaser.arriveAndAwaitAdvance();
                                  task.run();
                                  }).start();
                });

   phaser.arriveAndDeregister();
 }
```

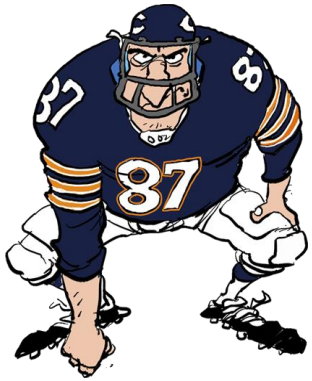
Create/start a new worker thread that runs the task once other threads arrive

Applying a One-shot Phaser with Java

- A one-shot Phaser that starts running a list of tasks simultaneously

```
void runTasks(List<MyTask> tasks) {
  Phaser phaser = new Phaser(1);

  tasks.forEach(task -> {
                phaser.register();

                new Thread(() -> {
                           phaser.arriveAndAwaitAdvance();
                           task.run();
                           }).start();
                });

   phaser.arriveAndDeregister();
 }
```

Block until all worker threads have started



Applying a One-shot Phaser with Java

- A one-shot Phaser that starts running a list of tasks simultaneously

```
void runTasks(List<MyTask> tasks) {
  Phaser phaser = new Phaser(1);

  tasks.forEach(task -> {
                phaser.register();

                new Thread(() -> {
                          phaser.arriveAndAwaitAdvance();
                          task.run();
                          }).start();
                });

   phaser.arriveAndDeregister();
 }
```

Block until all worker threads have started

This code is using the phaser as a one-shot "entry barrier"

Applying a One-shot Phaser with Java

- A one-shot Phaser that starts running a list of tasks simultaneously

```
void runTasks(List<MyTask> tasks) {
  Phaser phaser = new Phaser(1);

  tasks.forEach(task -> {
                phaser.register();

                new Thread(() -> {
                           phaser.arriveAndAwaitAdvance();
                           task.run();
                           }).start();
                });

   phaser.arriveAndDeregister();
 }
```

Run the task

Applying a One-shot Phaser with Java

- A one-shot Phaser that starts running a list of tasks simultaneously

```
void runTasks(List<MyTask> tasks) {
  Phaser phaser = new Phaser(1);

  tasks.forEach(task -> {
                phaser.register();

                new Thread(() -> {
                           phaser.arriveAndAwaitAdvance();
                           task.run();
                           }).start();
                });

   phaser.arriveAndDeregister();
 }
```

Allow thread calling runTasks() to continue & deregister itself so worker threads can start running their tasks

Applying a Cyclic Phaser with Java

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
              do { task.run(); phaser.arriveAndAwaitAdvance();
              } while (!phaser.isTerminated());
              }).start();
  });

  phaser.arriveAndDeregister();
}
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex26

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
                do { task.run(); phaser.arriveAndAwaitAdvance();
                } while (!phaser.isTerminated());
                }).start();
  });

  phaser.arriveAndDeregister();
}
```

Create a new phaser that iterates a given # of times

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
              do { task.run(); phaser.arriveAndAwaitAdvance();
              } while (!phaser.isTerminated());
              }).start();
  });

  phaser.arriveAndDeregister();
}
```

This hook method determines when to terminate the phaser

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
              do { task.run(); phaser.arriveAndAwaitAdvance();
              } while (!phaser.isTerminated());
              }).start();
  });

  phaser.arriveAndDeregister();
}
```

This phaser terminates when all iterations have completed

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
                do { task.run(); phaser.arriveAndAwaitAdvance();
                } while (!phaser.isTerminated());
                }).start();
  });

  phaser.arriveAndDeregister();
}
```

Register to defer worker threads advancing to second phase until the end of this method

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
                do { task.run(); phaser.arriveAndAwaitAdvance();
                } while (!phaser.isTerminated());
                }).start();
  });

  phaser.arriveAndDeregister();
}
```

Iterate thru all the tasks

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
                do { task.run(); phaser.arriveAndAwaitAdvance();
                } while (!phaser.isTerminated());
                }).start();
  });

  phaser.arriveAndDeregister();
}
```

Inform phaser there's a new party to add

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
                do { task.run(); phaser.arriveAndAwaitAdvance();
                } while (!phaser.isTerminated());
                }).start();
  });

  phaser.arriveAndDeregister();
}
```

Create/start a new worker thread & run the task

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
                do { task.run(); phaser.arriveAndAwaitAdvance();
                } while (!phaser.isTerminated());
                }).start();
  });

  phaser.arriveAndDeregister();
}
```

Block until all other tasks/ threads complete this phase

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
                do { task.run(); phaser.arriveAndAwaitAdvance();
                } while (!phaser.isTerminated());
                }).start();
  });

  phaser.arriveAndDeregister();
}
```

Block until all other tasks/ threads complete this phase

This code is using the phaser as a "cyclic exit barrier"

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
                do { task.run(); phaser.arriveAndAwaitAdvance();
                } while (!phaser.isTerminated());
                }).start();
  });

  phaser.arriveAndDeregister();
}
```

The last thread to arrive at the end of a phase triggers the invocation of the onAdvance() hook method

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
                do { task.run(); phaser.arriveAndAwaitAdvance();
                } while (!phaser.isTerminated());
                }).start();
  });

  phaser.arriveAndDeregister();
}
```

The phaser is terminated when the phase # is >= the iterations param

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
                do { task.run(); phaser.arriveAndAwaitAdvance();
                } while (!phaser.isTerminated());
                }).start();
  });

  phaser.arriveAndDeregister();
}
```

Loop until phaser is terminated by onAdvance()

Applying Cyclic Phaser with Java

- A cyclic Phaser that repeatedly performs actions for a given # of iterations

```
void startTasks(List<MyTask> tasks, int iterations) {
  Phaser phaser = new Phaser() {
    protected boolean onAdvance(int phase, int regParties)
    { return phase >= iterations || regParties == 0; }
  };

  phaser.register();

  tasks.forEach(task -> {
    phaser.register();
    new Thread(() -> {
                do { task.run(); phaser.arriveAndAwaitAdvance();
                } while (!phaser.isTerminated());
                }).start();
  });

  phaser.arriveAndDeregister();
}
```

Deregister itself (allowing tasks to advance to next phase) & don't wait

End of Java Phaser: Example Application