```java
public class MatrixMulTask extends RecursiveAction {
  static final int THRESHOLD = ...;
  Matrix lhs, rhs, product;
  public MatrixMulTask(Matrix lhs, Matrix rhs, Matrix product) {
    this.lhs = lhs;
    this.rhs = rhs;
    this.product = product;
  }
  public void compute() {
    int n = lhs.getDim();
    if (n <= THRESHOLD) {
      Matrix.multiply(lhs, rhs, product);
    } else {
      List<MatrixMulTask> tasks = new ArrayList<>(8);
      Matrix[] term = new Matrix[]{new Matrix(n), new Matrix(n)};
      for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
          for (int k = 0; k < 2; k++) {
            tasks.add(
                    new MatrixMulTask(
                            lhs.split(j, i),
                            rhs.split(i, k),
                            term[i].split(j, k)
                    )
            );
          }
        }
      }
      tasks.stream().forEach((task) -> {
        task.fork();
      });
      tasks.stream().forEach((task) -> {
        task.join();
      });
      (new MatrixAddTask(term[0], term[1], product)).compute();
    }
  }
}
```

FIGURE 16.6 The MatrixMulTask class: fork-join parallel matrix addition.