**Question 5**

**Multithreaded Java Application**

**Develop a Java Based multithreaded application to process six tasks (A, B, C, D, E and F) with the following execution flow:**

**Taks A & B should run parallel.**

**Task C & D should run in parallel after A & B complete**

**The output of A & B should be fed into C & D**

**The output of C & D should be fed into F & final processing**

**Design the application to handle success and failure modes for each task to ensure robustness and reliability.**

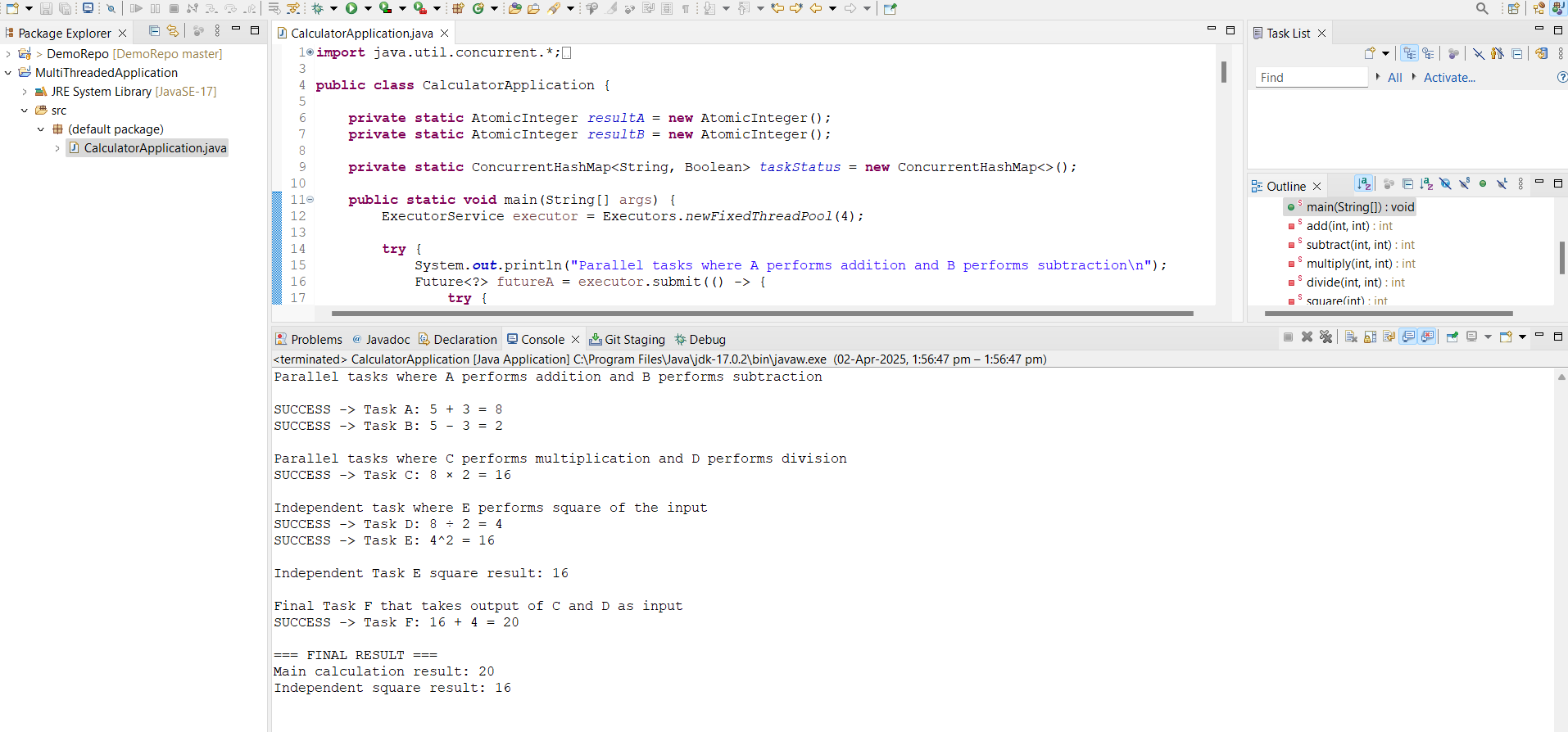
Created a simple Calculator Application that supports Parallel Task Processing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Operation | Description | Input | Output |
| A | Addition | Adds two numbers | None | 5 + 3 = 8 |
| B | Subtraction | Subtracts two numbers | None | 5 - 3 = 2 |
| C | Multiplication | Multiplies results from A & B | A's output (8), B's output (2) | 8 × 2 = 16 |
| D | Division | Divides result from A by result from B | A's output (8), B's output (2) | 8 ÷ 2 = 4 |
| E | Square (independent) | Squares a number | None | 4² = 16 |
| F | Final Calculation | Combines results from C & D | C's output (16), D's output (4) | 16 + 4 = 20 |

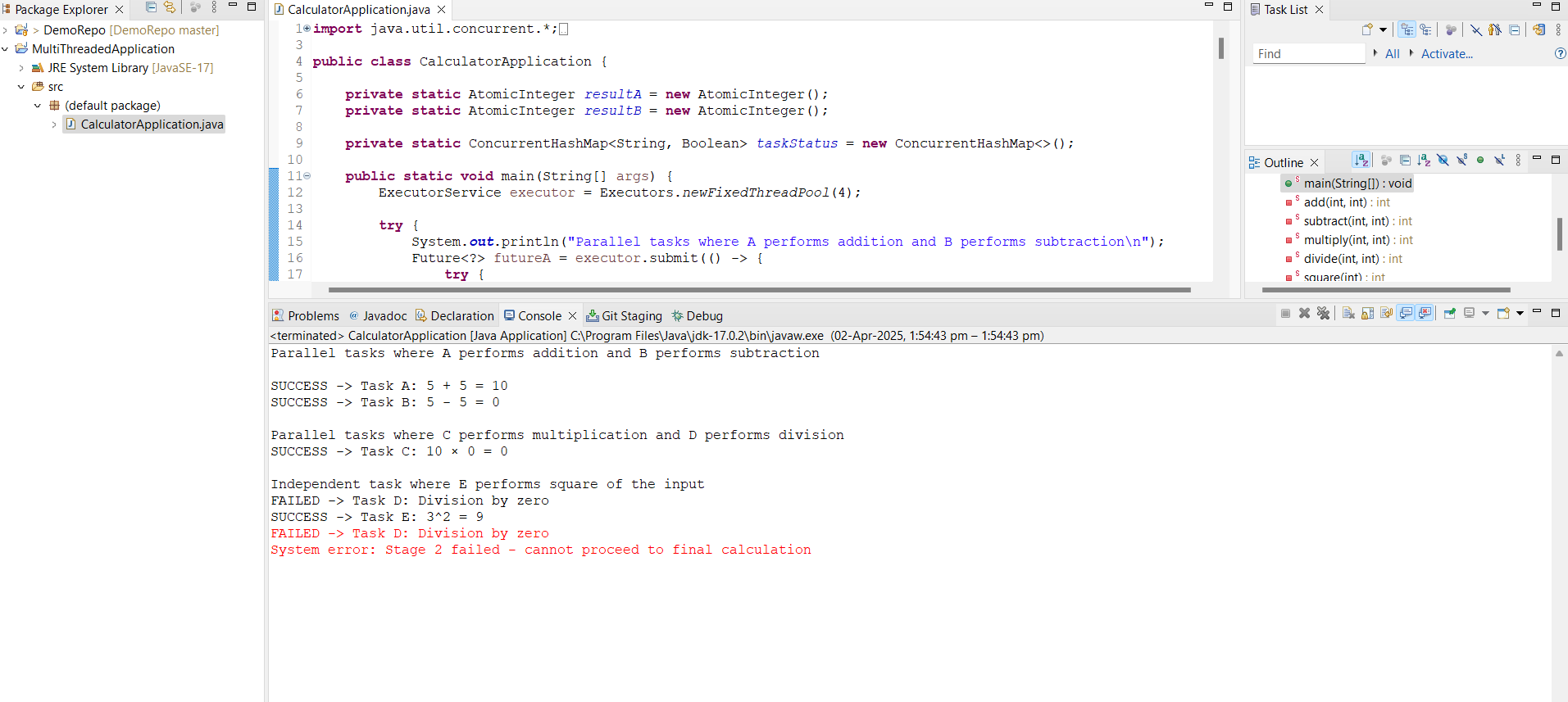
* **Try-Catch blocks for Math Operations**: Handles overflow and divide-by-zero errors and any other failures
* **AtomicInteger for Thread Safety**: Ensures safe updates to shared results and since it leverages hardware support its faster and it helps prevent race conditions.
* **ConcurrentHashMap for Task Tracking**: As it is thread safe and it keeps track of task statuses (success/failure) and allows early termination if prerequisites fail.
* **Future & Async Processing**: Used Future for asynchronous task and .get() waits for completion, and ExecutorService manages a fixed thread pool (4 threads).
* **Timeout Handling**: getWithTimeout() prevents indefinite waits, failing fast if tasks exceed 2 seconds.

**Output:**

**Success scenario**

****

**Failure scenario**

****