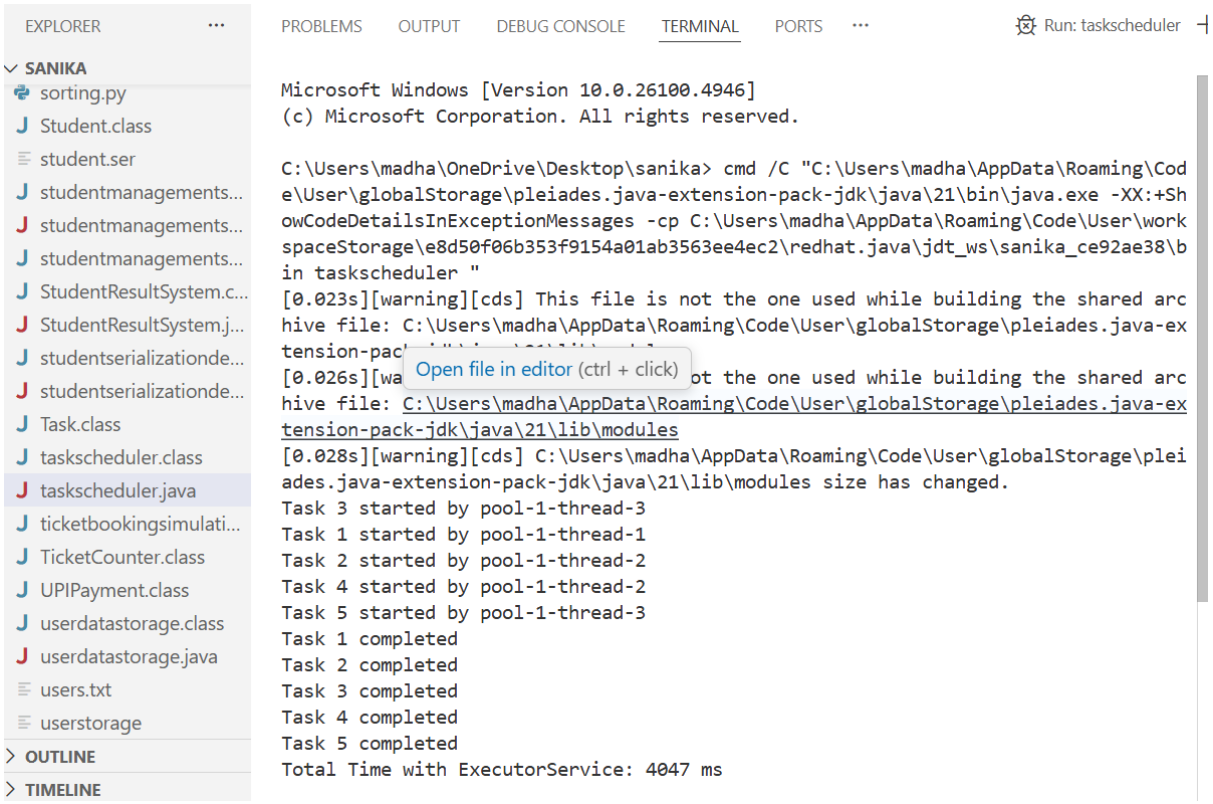
```
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.TimeUnit;

class Task implements Callable<String> {
    private int taskId;

    public Task(int taskId) {
        this.taskId = taskId;
    }

    @Override
    public String call() throws Exception {
        System.out.println("Task " + taskId + " started by " + Thread.currentThread().getName());
        Thread.sleep(2000); // simulate work
        return "Task " + taskId + " completed";
    }

    public void setTaskId(int taskId) {
        this.taskId = taskId;
    }
}

public class taskscheduler {

    public static void main(String[] args) {

        // Create fixed thread pool of size 3
```

```
ExecutorService executor = Executors.newFixedThreadPool(3);

long startTime = System.currentTimeMillis();

Future<String>[] futures = new Future[5];

// Submit 5 tasks
for (int i = 0; i < 5; i++) {
    futures[i] = executor.submit(new Task(i + 1));
}

// Monitor task execution
for (int i = 0; i < 5; i++) {
    try {
        System.out.println(futures[i].get());
    } catch (InterruptedException | ExecutionException e) {
        System.out.println("Task failed: " + e.getMessage());
    }
}

// Graceful shutdown
executor.shutdown();
try {
    if (!executor.awaitTermination(5, TimeUnit.SECONDS)) {
        executor.shutdownNow();
    }
} catch (InterruptedException e) {
    executor.shutdownNow();
}

long endTime = System.currentTimeMillis();
```

```
        System.out.println("Total Time with ExecutorService: " + (endTime - startTime) + " ms");
    }
}
```




```java
// Source code is decompiled from a .class file using FernFlower decompiler (from Intellij
import java.util.concurrent.Callable;

class Task implements Callable<String> {
   private int taskId;

   public Task(int var1) {
      this.taskId = var1;
   }

   public String call() throws Exception {
      int var10001 = this.taskId;
      System.out.println("Task " + var10001 + " started by " + Thread.currentThread().getN
      Thread.sleep(2000L);
      return "Task " + this.taskId + " completed";
   }

   public void setTaskId(int var1) {
      this.taskId = var1;
   }
}
```