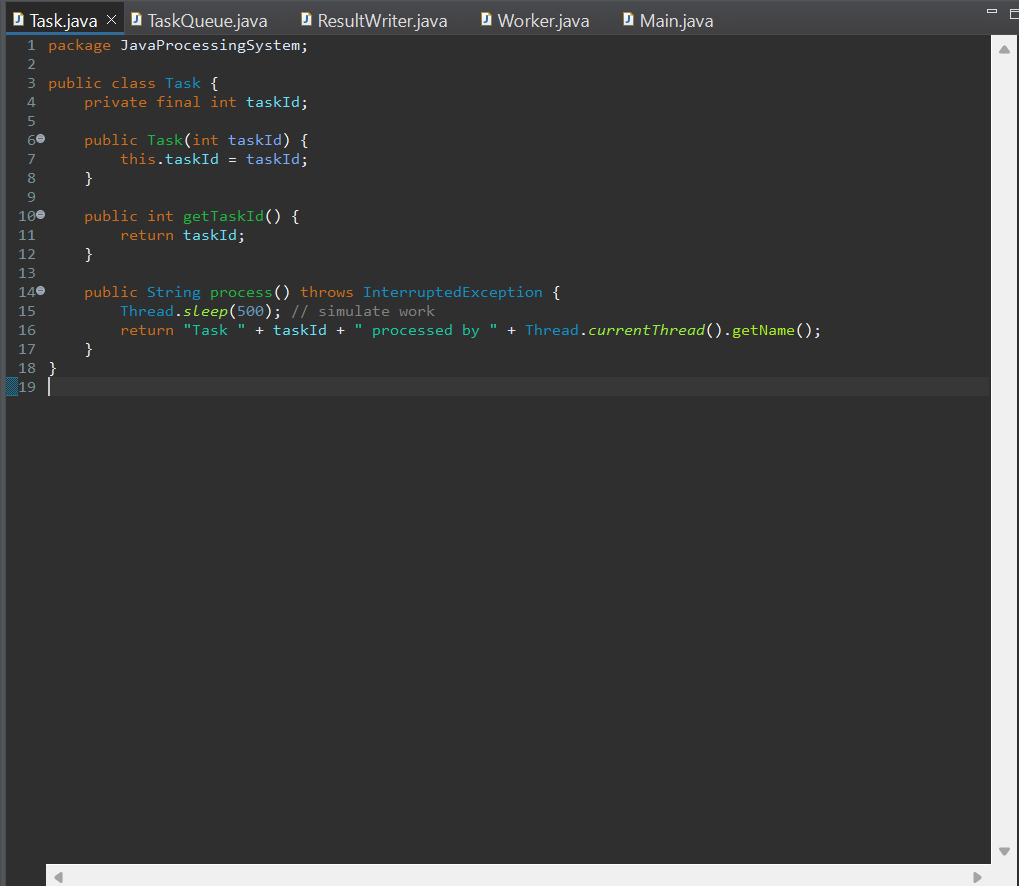
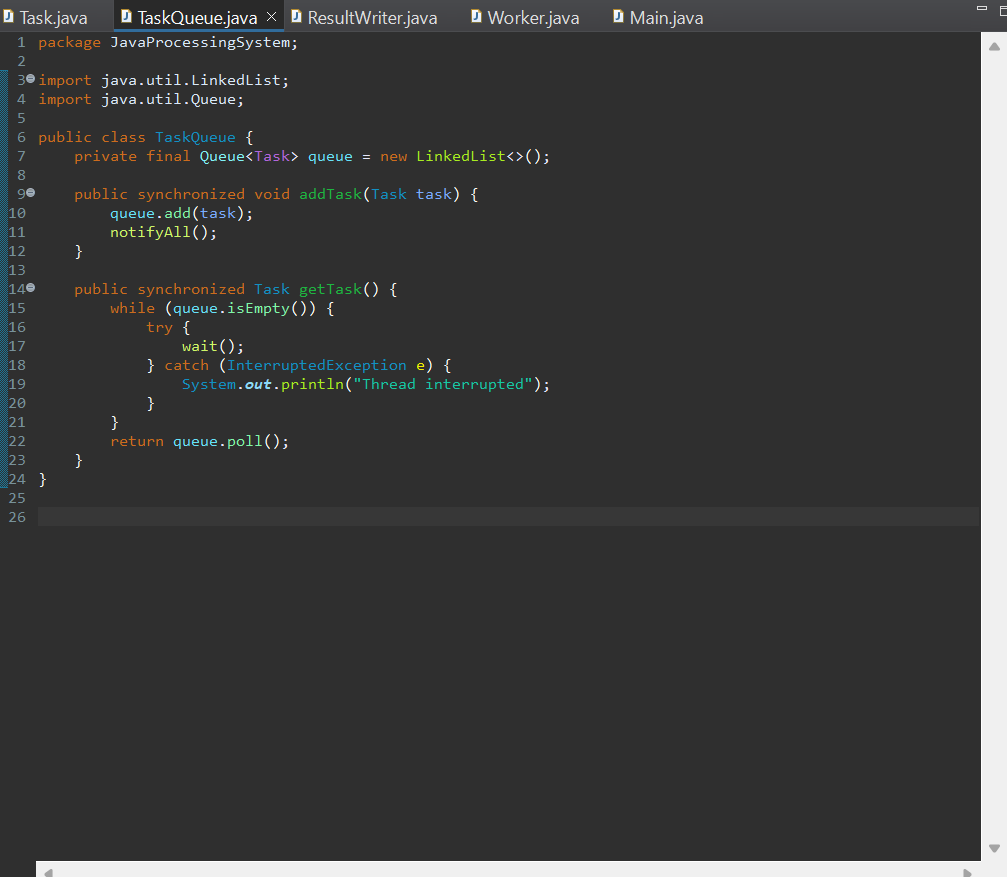
**1. Introduction**

The Multi-threaded Data Processing System is designed to simulate a ride-sharing platform where multiple worker threads or goroutines retrieve tasks from a shared queue, process them, and save the results. This system demonstrates effective use of concurrency and synchronization to manage data processing in parallel, ensuring efficient task completion and avoiding common issues such as race conditions or deadlocks. The project is implemented in both Java and Go to illustrate different concurrency models and error-handling techniques.

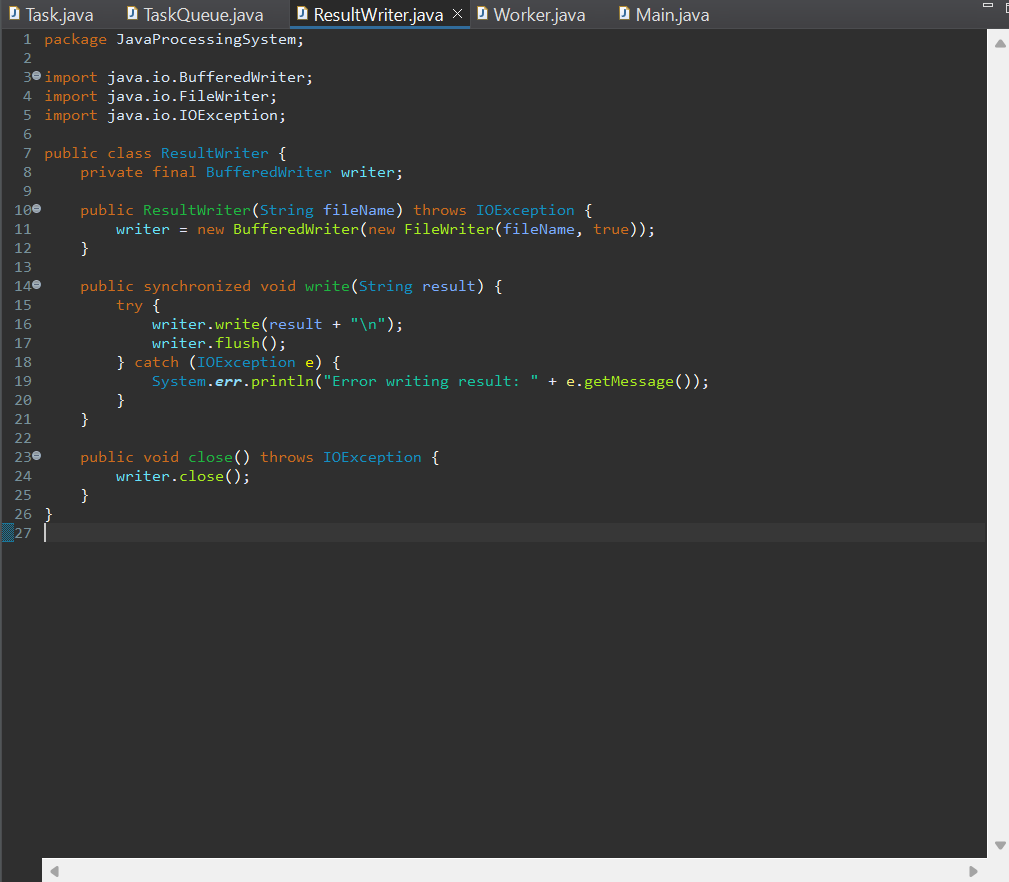
**Java Code**

**Task.java**

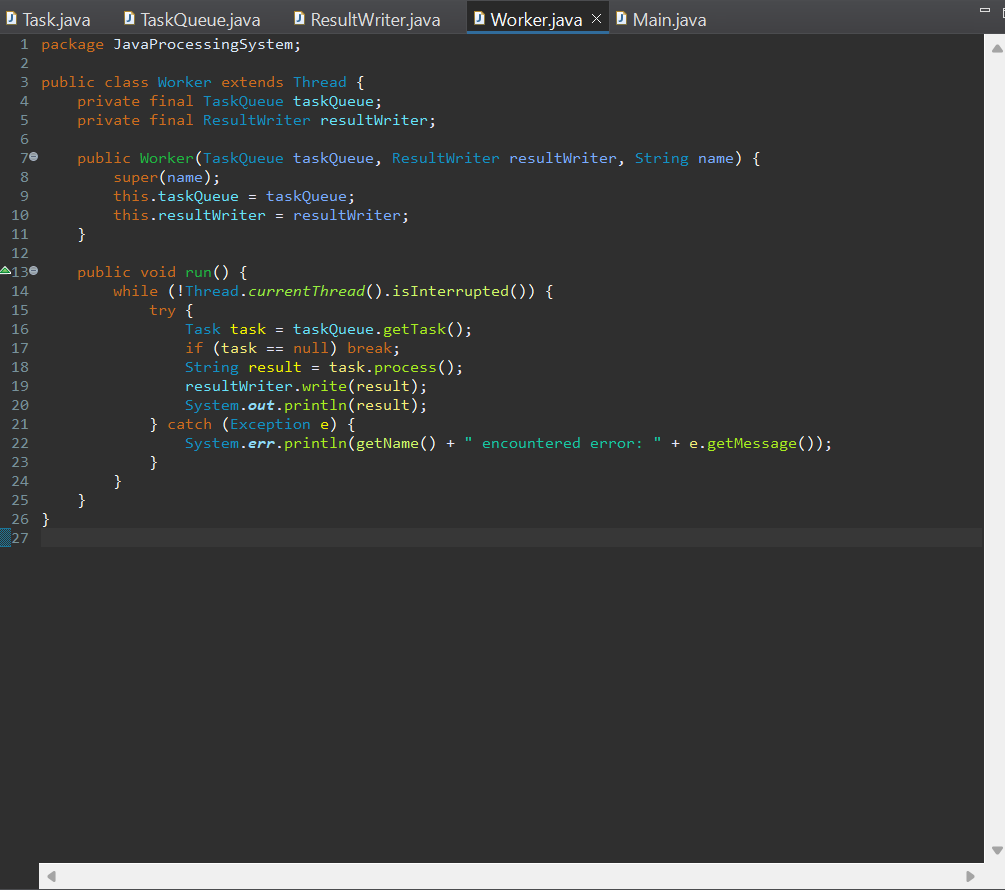
****

TaskQueue.java  
****

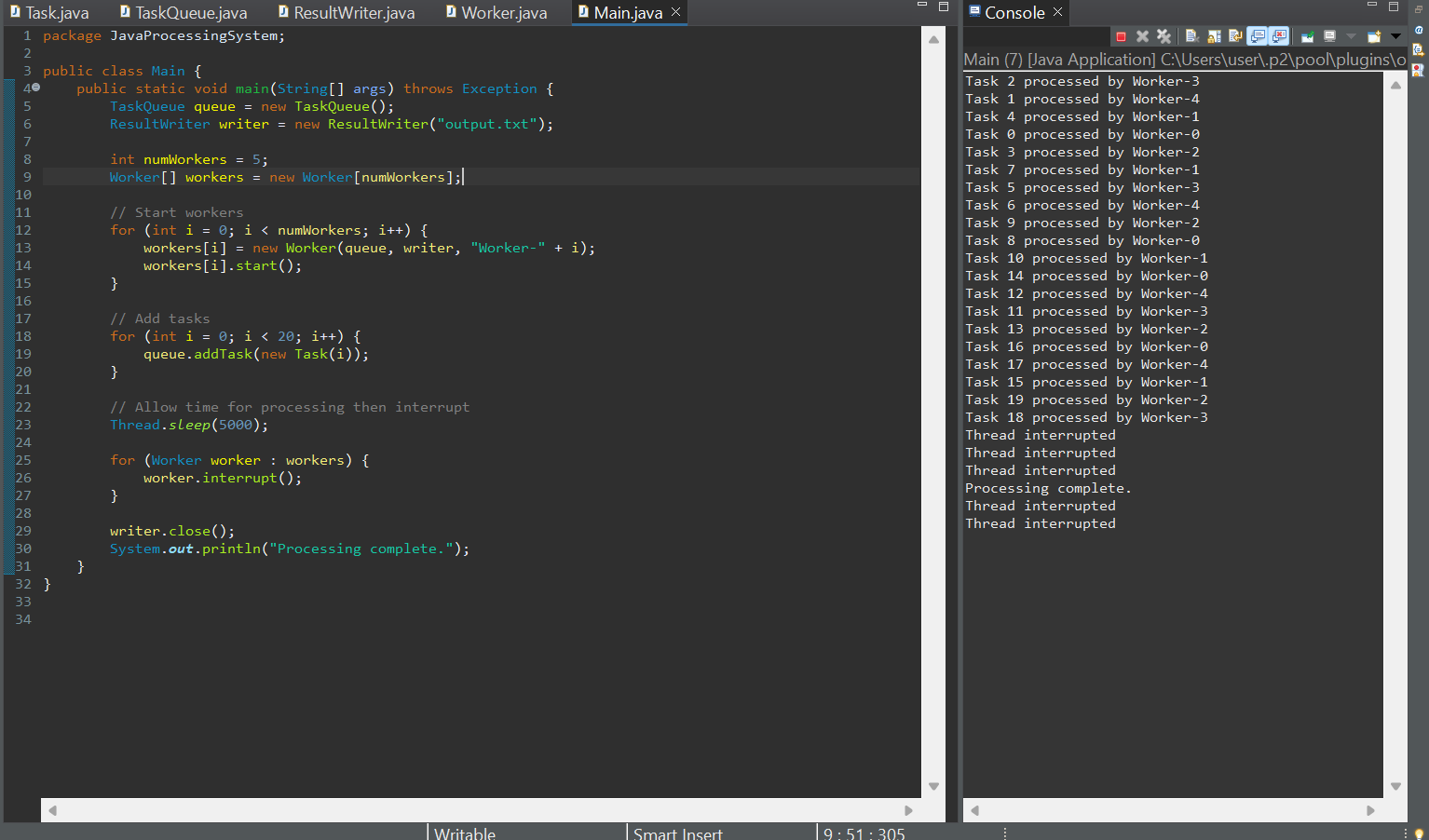
ResultWriter.java

****

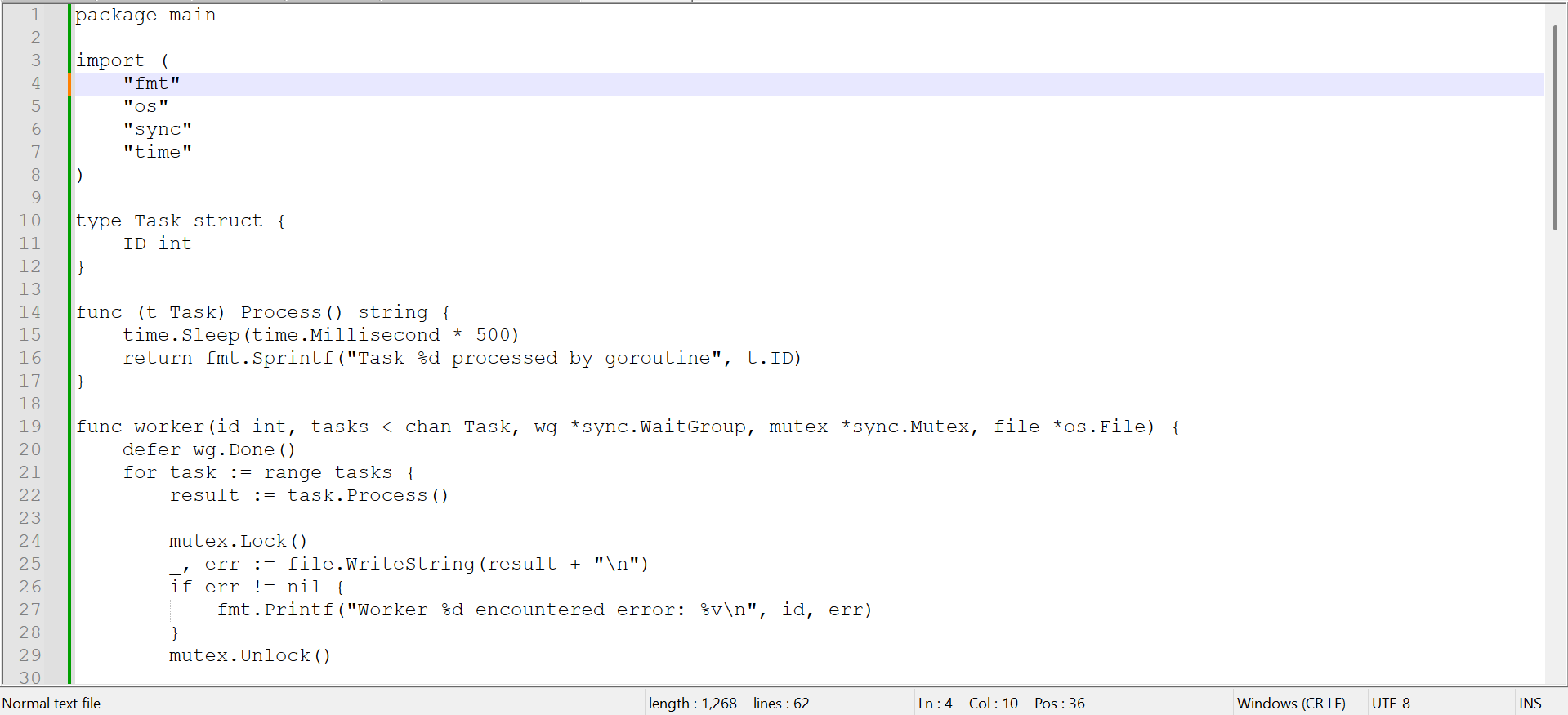
Worker.java

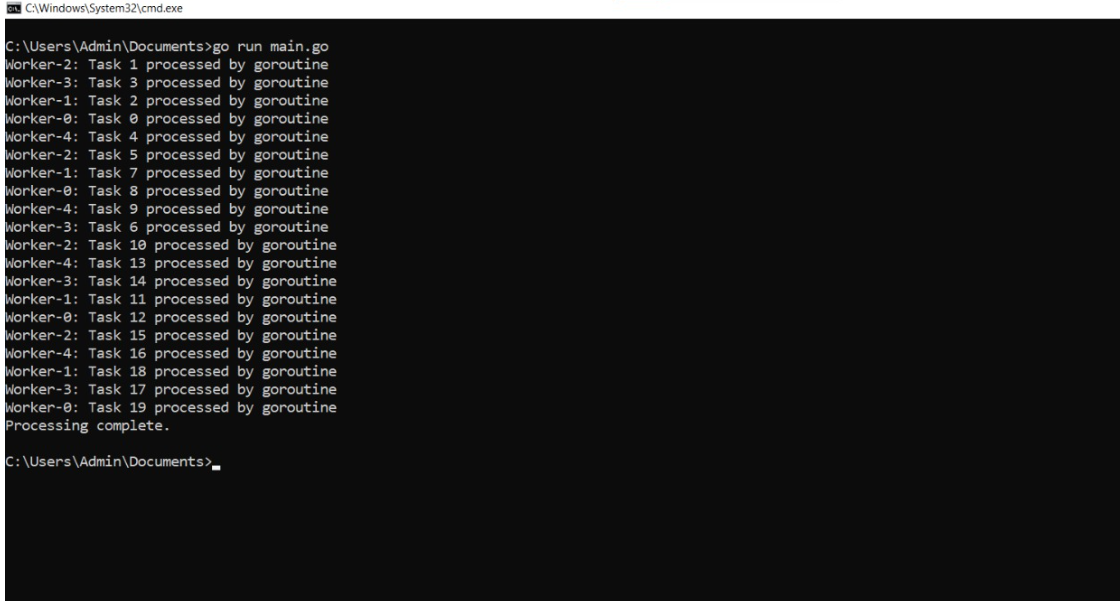
****

Main.java

****

Implement in Go:

  
  
output:

****

**Concurrency Techniques Used**

**Java:**

Used Thread class to create multiple worker threads.  
 Synchronization of task queue using synchronized blocks and wait/notify mechanism.  
 Each worker retrieves a task, simulates processing using sleep, and writes results to a shared file using synchronized file write operations.

**Go:**

Utilized goroutines for concurrent worker execution.  
 Channels used as thread-safe queues to distribute tasks.  
 Mutex lock used to prevent race conditions while writing results to a shared file.  
 WaitGroup ensures all goroutines complete before the program exits.

**Exception Handling**

**Java:**

try-catch blocks used to handle exceptions such as InterruptedException during thread sleep or IOExceptions during file operations.  
 Thread interruption handled gracefully with proper resource cleanup.

**Go:**

Errors are handled by checking returned values from functions like os.Create and file.WriteString.  
 defer statements used for clean-up operations like closing files.

**Comparison:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Java** | **Go** |
| Threading Model | Threads via Thread class | Goroutines (lightweight) |
| Synchronization | synchronized, wait/notify | Channels, sync.Mutex |
| Error Handling | try-catch | Error values + defer |
| Logging | Console/File | Console/File |