How to create your own thread pool in Java – 2022

import java.util.ArrayList;  
import java.util.List;  
import java.util.concurrent.BlockingQueue;  
import java.util.concurrent.LinkedBlockingQueue;  
  
public class CustomThreadPool {  
  
 *// holds tasks***private BlockingQueue<Runnable> runnableQueue;**  
 *// holds the pool of worker threads* **private List<WorkerThread> threads;**  
 *// check if shutdown is initiated* private boolean isShutDown;  
  
 public CustomThreadPool(final int noOfThreads) {  
 this.runnableQueue = new LinkedBlockingQueue<>();  
 this.threads = new ArrayList<>(noOfThreads);  
 this.isShutDown = false;  
 *// create worker threads* **for (int i = 1; i <= noOfThreads; i++) {  
 WorkerThread thread = new WorkerThread(runnableQueue, this);  
 thread.setName("Worker Thread - " + i);  
 thread.start();  
 threads.add(thread);  
 }** }  
  
 public void execute(Runnable r) throws InterruptedException {  
 **if (!isShutDown) {  
 runnableQueue.put(r);  
 } else {  
 throw new InterruptedException("Thread Pool shutdown is initiated, unable to execute task");  
 }** }  
  
 **public void shutdown() {  
 isShutDown = true;  
 }**

**private class WorkerThread extends Thread** {  
 *// holds tasks* **private BlockingQueue<Runnable> runnableQueue;**  
 *// check if shutdown is initiated***private CustomThreadPool threadPool**;  
  
 **public WorkerThread(BlockingQueue<Runnable> taskQueue, CustomThreadPool threadPool) {  
 this.runnableQueue = taskQueue;  
 this.threadPool = threadPool;  
 }**  
 @Override  
 public void run() {  
 try {  
 *// continue until all tasks finished processing* **while (!threadPool.isShutDown || !runnableQueue.isEmpty()) {  
 Runnable runnable;  
 *// Poll a runnable from the queue and execute it* while ((runnable = runnableQueue.poll()) != null) {  
 runnable.run();  
 }  
 Thread.*sleep*(1);  
 }** } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
}

**Test Class**

public class CustomThreadPoolTest {  
  
 public static void main(String[] args) throws InterruptedException {  
 Runnable r = () -> {  
 try {  
 Thread.*sleep*(5000);  
 System.*out*.println(Thread.*currentThread*().getName() + " is executing task.");  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 };  
  
 CustomThreadPool threadPool = new CustomThreadPool(2);  
  
 threadPool.execute(r);  
 threadPool.execute(r);  
 threadPool.shutdown();  
 }  
  
}