```
--------------BlockingQueueTask.java---------
package com.ameya.tasks;

public class BlockingQueueTask implements Runnable {

    private String name=null;
    public BlockingQueueTask(String name) {
        this.name=name;
    }
    public String getName() {
        return name;
    }
    @Override
    public void run() {
        try {
            Thread.sleep(500);
        }catch(InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Thread :: "+name);
    }

}
--------------CustomThreadPoolExecutor.java--------------------
-
package com.ameya.executors;

import java.util.concurrent.BlockingQueue;
import java.util.concurrent.RejectedExecutionHandler;
import java.util.concurrent.ThreadFactory;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;

public class CustomThreadPoolExecutor extends ThreadPoolExecutor {

    public CustomThreadPoolExecutor(int corePoolSize, int
    maximumPoolSize, long keepAliveTime, TimeUnit unit,
            BlockingQueue<Runnable> workQueue) {
        super(corePoolSize, maximumPoolSize, keepAliveTime, unit,
        workQueue);
    }
```

```java

    @Override
    protected void beforeExecute(Thread t, Runnable r) {
        super.beforeExecute(t, r);
        System.out.println("beforeExecute logic done");
    }

    @Override
    protected void afterExecute(Runnable r, Throwable t) {
        super.afterExecute(r, t);
        if(t!=null) {
            System.out.println("Exception Handler Logic Done");
        }
        System.out.println("afterExecute logic done");
    }

}
-------------------TestBlockingQueue.java---------
package com.ameya.test;

import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.RejectedExecutionHandler;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;

import com.ameya.executors.CustomThreadPoolExecutor;
import com.ameya.tasks.BlockingQueueTask;

public class TestBlockingQueue {

    public static void main(String[] args) {
        Integer threadCnt=0;
        BlockingQueue<Runnable> worksQueue=new
        ArrayBlockingQueue<Runnable>(50);
        CustomThreadPoolExecutor executor=new
        CustomThreadPoolExecutor(10, 20, 5000,
        TimeUnit.MILLISECONDS, worksQueue);
        executor.setRejectedExecutionHandler(
                new RejectedExecutionHandler() {

                    @Override
                    public void rejectedExecution(Runnable r, ThreadPoolExecutor executor) {
                        System.out.println("TASK REJECTED : "+((BlockingQueueTask) r).getName());
                        System.out.println("WAITING FOR SECOND..");
                        try {
                            Thread.sleep(1000);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                        System.out.println("ANOTHER TIME : "+((BlockingQueueTask) r).getName());
                        executor.execute(r);
                    }});
        executor.prestartAllCoreThreads();

        while(true) {
            threadCnt++;
            System.out.println("Adding TAsk : "+threadCnt);
            executor.execute(new BlockingQueueTask(threadCnt+""));
            if(threadCnt==100)
                break;
        }

    }

}
```