Main.java

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;


public class Main
{

	public static void main(String[] args) throws InterruptedException
	{
		// TODO Auto-generated method stub
		MyPool myPool = new MyPool(20);
		ExecutorService threadPool = Executors.newFixedThreadPool(2);
		MyThread t1 = new MyThread("A", myPool, 3);
		MyThread t2 = new MyThread("B", myPool, 12);
		MyThread t3 = new MyThread("C", myPool, 7);
		threadPool.execute(t1);
		threadPool.execute(t2);
		threadPool.execute(t3);
		threadPool.shutdown();
		while (false == threadPool.isTerminated())
		{
			Thread.sleep(100);
		}
		System.out.println("Done");
	}

}
```

MyPool.java

```
import java.util.concurrent.Semaphore;

public class MyPool
{
	private Semaphore sp;

	MyPool (int size)
	{
		this.sp = new Semaphore(size);
	}

	public Semaphore getSp ()
	{
		return this.sp;
	}

	public void setSp (Semaphore sp)
	{
		this.sp = sp;
	}


}
```

MyThread.java

```
public class MyThread extends Thread
{
	private String threadname;
	private MyPool pool;
	private int x;

	MyThread (String threadname, MyPool pool, int x)
	{
		this.threadname = threadname;
		this.pool = pool;
		this.x = x;
	}

	public void run ()
	{
		try
		{
			Thread.sleep(2000);
		} catch (InterruptedException e1)
		{
			// TODO Auto-generated catch block
			e1.printStackTrace();
		}
		try
		{
			pool.getSp().acquire(x);
			System.out.println(threadname + "成功获取了" + x + "个许可！");
		} catch (Exception e)
		{
			// TODO: handle exception
			e.printStackTrace();
		}finally{
			pool.getSp().release();
			System.out.println(threadname + "成功释放了" + x + "个许可！");
		}
	}

}
```