实验一参考代码

- 任务一

参考代码：

```
public class AlternatePrinting {

    // 共享对象，用于线程同步
    private static final Object lock = new Object();
    private static boolean isLetterTurn = true; // 控制字母线程的执行

    public static void main(String[] args) {
        // 创建字母线程（通过实现 Runnable 接口）
        Thread letterThread = new Thread(new LetterPrinter(),
"LetterThread");

        // 创建数字线程（通过继承 Thread 类）
        Thread numberThread = new NumberPrinter("NumberThread");

        // 启动线程
        letterThread.start();
        numberThread.start();
    }

    // 字母打印线程（实现 Runnable 接口）
    static class LetterPrinter implements Runnable {
        private final char[] letters = {'a', 'b', 'c', 'd', 'e'};

        @Override
        public void run() {
            for (char letter : letters) {
                synchronized (lock) {
                    // 如果不是字母线程的回合，则等待
                    while (!isLetterTurn) {
                        try {
                            lock.wait();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                    // 打印字母
                    System.out.print(letter);
                    isLetterTurn = false; // 切换到数字线程
                    lock.notify(); // 唤醒数字线程
                }
```

```
            }
        }
    }

    // 数字打印线程（继承 Thread 类）
    static class NumberPrinter extends Thread {
        private final int[] numbers = {1, 23, 456, 7891, 23456};

        public NumberPrinter(String name) {
            super(name);
        }

        @Override
        public void run() {
            for (int number : numbers) {
                synchronized (lock) {
                    // 如果不是数字线程的回合，则等待
                    while (isLetterTurn) {
                        try {
                            lock.wait();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                    // 打印数字
                    System.out.print(number);
                    isLetterTurn = true; // 切换到字母线程
                    lock.notify(); // 唤醒字母线程
                }
            }
        }
    }
}
```

输出结果：

```
a1b23c456d7891e23456
```

- 任务二

```
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import java.util.concurrent.locks.Condition;

class TicketPool {
    private int tickets; // 余票数量
    private Lock lock = new ReentrantLock();
    private Condition condition = lock.newCondition();
```

```
9
10      public TicketPool(int initialTickets) {
11          System.out.println("初始所有窗口票数为: " + initialTickets);
12
13          this.tickets = initialTickets;
14      }
15
16      // 售票操作
17      public void sellTickets(int windowId, int numTickets) {
18          lock.lock();
19          try {
20              while (tickets < numTickets) {
21                  System.out.println("窗口 " + windowId + " 余票不足，购票者等
待...");
22                  condition.await(); // 等待新票进入
23              }
24              tickets -= numTickets;
25              System.out.println("窗口 " + windowId + " 售出 " + numTickets +
" 张票，余票: " + tickets);
26          } catch (InterruptedException e) {
27              e.printStackTrace();
28          } finally {
29              lock.unlock();
30          }
31      }
32
33      // 退票操作
34      public void returnTickets(int windowId, int numTickets) {
35          lock.lock();
36          try {
37              tickets += numTickets;
38              System.out.println("窗口 " + windowId + " 退票 " + numTickets +
" 张，余票: " + tickets);
39              condition.signalAll(); // 通知所有等待的购票者
40          } finally {
41              lock.unlock();
42          }
43      }
44
45      // 新进票操作
46      public void addTickets(int numTickets) {
47          lock.lock();
48          try {
49              tickets += numTickets;
50              System.out.println("新进票 " + numTickets + " 张，余票: " +
tickets);
51              condition.signalAll(); // 通知所有等待的购票者
52          } finally {
```

```
            lock.unlock();
        }
    }

    public int getTickets() {
        return tickets;
    }
}

// 售票窗口线程
class TicketWindow implements Runnable {
    private TicketPool ticketPool;
    private int windowId;

    public TicketWindow(TicketPool ticketPool, int windowId) {
        this.ticketPool = ticketPool;
        this.windowId = windowId;
    }

    @Override
    public void run() {
        while (true) {
            try {
                // 模拟随机售票或退票操作
                if (Math.random() > 0.5) {
                    int numTickets = (int) (Math.random() * 5) + 1; // 随机购买1-5张票
                    ticketPool.sellTickets(windowId, numTickets);
                } else {
                    int numTickets = (int) (Math.random() * 3) + 1; // 随机退1-3张票
                    ticketPool.returnTickets(windowId, numTickets);
                }
                Thread.sleep(1000); // 模拟操作间隔
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

// 新进票线程
class AddTicketTask implements Runnable {
    private TicketPool ticketPool;

    public AddTicketTask(TicketPool ticketPool) {
        this.ticketPool = ticketPool;
    }
```

```
    @Override
    public void run() {
        while (true) {
            try {
                Thread.sleep(5000); // 每5秒新进一次票
                int numTickets = (int) (Math.random() * 10) + 5; // 随机新进5-15张票
                ticketPool.addTickets(numTickets);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class TicketSystem {
    public static void main(String[] args) {
        TicketPool ticketPool = new TicketPool(20); // 初始票数为20

        // 创建3个售票窗口线程
        for (int i = 1; i <= 3; i++) {
            new Thread(new TicketWindow(ticketPool, i)).start();
        }

        // 创建新进票线程
        new Thread(new AddTicketTask(ticketPool)).start();
    }
}

```

输出效果：

```
初始所有窗口票数为: 20
窗口 2 退票 1 张，余票: 21
窗口 1 退票 2 张，余票: 23
窗口 3 退票 3 张，余票: 26
窗口 3 退票 3 张，余票: 29
窗口 1 退票 2 张，余票: 31
窗口 2 退票 1 张，余票: 32
窗口 2 退票 3 张，余票: 35
窗口 1 退票 2 张，余票: 37
窗口 3 退票 3 张，余票: 40
窗口 3 退票 2 张，余票: 42
窗口 1 售出 5 张票，余票: 37
窗口 2 售出 5 张票，余票: 32
窗口 2 退票 1 张，余票: 33
窗口 3 退票 3 张，余票: 36
```

```
窗口 1 退票 2 张，余票：38
新进票 11 张，余票：49
窗口 3 售出 3 张票，余票：46
窗口 1 退票 1 张，余票：47
窗口 2 售出 4 张票，余票：43
窗口 1 退票 2 张，余票：45
窗口 2 售出 1 张票，余票：44
窗口 3 售出 1 张票，余票：43
窗口 2 退票 2 张，余票：45
窗口 1 售出 5 张票，余票：40
窗口 3 退票 1 张，余票：41
窗口 2 退票 2 张，余票：43
窗口 1 售出 1 张票，余票：42
窗口 3 售出 3 张票，余票：39
窗口 1 售出 4 张票，余票：35
窗口 2 退票 3 张，余票：38
窗口 3 售出 3 张票，余票：35
```