哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2019 年春季学期

计算机学院《软件构造》课程

Lab 6 实验报告

姓名	
学号	
班号	
电子邮件	
手机号码	

目录

1 实验目标概述

本次实验训练学生的并行编程的基本能力，特别是 Java 多线程编程的能力。根据一个具体需求，开发两个版本的模拟器，仔细选择保证线程安全 (threadsafe) 的构造策略并在代码中加以实现，通过实际数据模拟，测试程序是否是线程安全 的。另外，训练学生如何在 threadsafe 和性能之间寻求较优的折中，为此计算吞 吐率和公平性等性能指标，并做仿真实验。

- Java 多线程编程
- 面向线程安全的 ADT 设计策略选择、文档化
- 模拟仿真实验与对比分析

2 实验环境配置

实验环境与 Lab5 完全相同

GitHub 仓库地址如下：

https://github.com/ComputerScienceHIT/Lab6-1173710229

3 实验过程

3.1 ADT 设计方案

1. Monkey

```
/**
 * class for Monkey,
 * this class represents the Monkey on the ladder,
 * and they are going to simulate passing river.
 */
```

2. Ladder

```
/**
 * this class stands for the ladder on the river.
 */
```

3. 接口 LadderSelector

```
/**
 * this interface specifies what a ladder selector
should be like.
```

```
 * they should have method select(), then pass in a
monkey and ladder list
 */
```

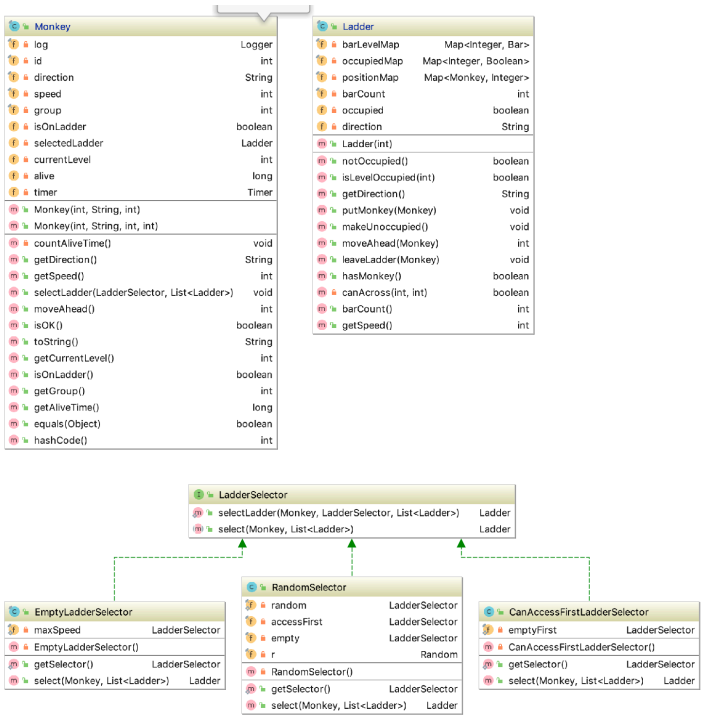
4. CanAccessFirstLadderSelector

```
/**
 * select the ladder when the direction
 * is the same and the first bar is available.
 */
```

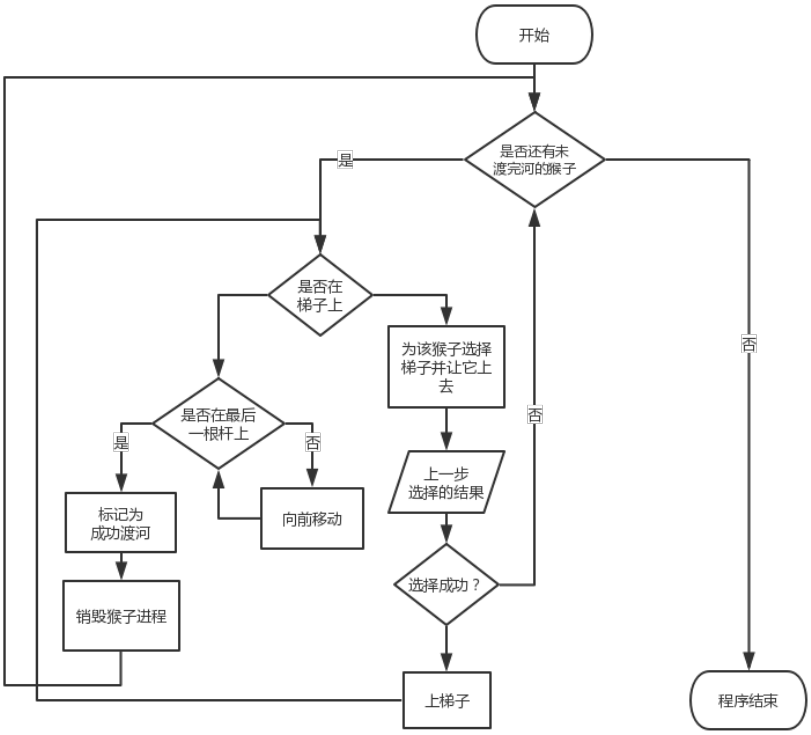
5. EmptyLadderSelector

```
/**
 * select the empty ladder.
 * i.e. when all ladders have monkey,
 * this selector will return null continuously.
 */
```

以下是 ADT 的类图和其之间的关系：



3.2 Monkey 线程的 run()的执行流程图



3.3 至少两种“梯子选择”策略的设计与实现方案

3.3.1 策略 1

选择方向相同、第一根杆子未被占用的梯子，实现代码如下：

```
@Override
public synchronized Ladder select(Monkey monkey, List<Ladder>
ladders) {
    String direction = monkey.getDirection();
    for (Ladder ladder : ladders) {
        if (ladder.notOccupied()) {
            return ladder;
        } else if (direction.equals(ladder.getDirection())
                && !ladder.isLevelOccupied(1)) {
            return ladder;
        }
    }
    return null;
}
```

调用方法时，如果希望避免对梯子的竞争，需要在调用前对梯子列表上锁。

3.3.2 策略 2

选择空梯子，意即如果所有梯子都不为空，猴子等待下一轮选择梯子的时机，实现代码如下：

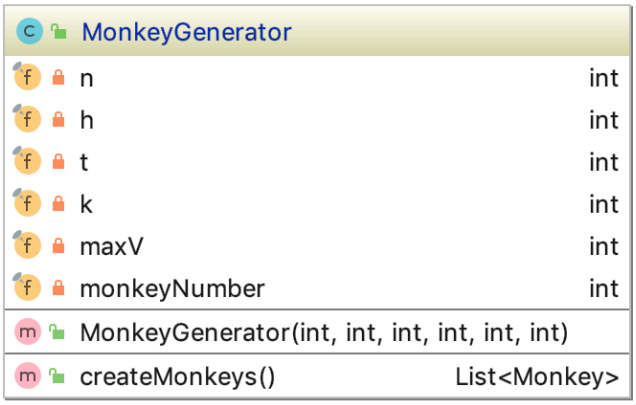
```
@Override
public Ladder select(Monkey monkey, List<Ladder> ladders)
{
    for (Ladder ladder : ladders) {
        if (ladder.notOccupied()) {
            return ladder;
        }
    }
    return null;
}
```

调用方法时，如果希望避免对梯子的竞争，需要在调用前对梯子列表上锁。

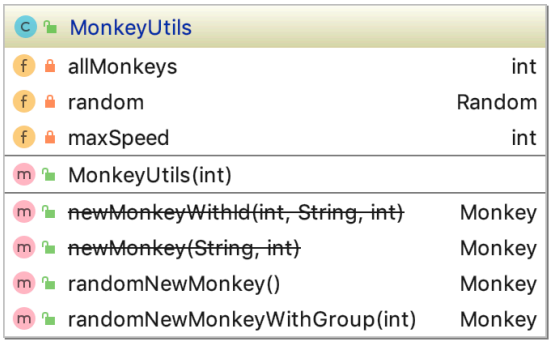
3.3.3 策略 3（可选）

3.4 “猴子生成器”MonkeyGenerator

MonkeyGenerator 的设计如图所示：



对象在构造的时候要求传入各个参数来决定如何生成猴子，生成猴子具体的操作交给 MonkeyUtils 中的 randomNewMonkeyWithGroup()方法，该类的设计如下：



randomNewMonkeyWithGroup()方法实现如下：

```
public Monkey randomNewMonkeyWithGroup(int group) {
    allMonkeys++;
    int speed = (Math.abs(random.nextInt()) % maxSpeed) + 1;
    int direction = Math.abs(random.nextInt());
    if (direction % 2 == 0) {
        return new Monkey(allMonkeys, Monkey.Direction.L2R, speed,
group);
    } else {
        return new Monkey(allMonkeys, Monkey.Direction.R2L, speed,
group);
    }
}
```

3.5 如何确保 threadsafe?

1. 梯子选择：每次猴子在选择梯子前（调用选择方法前），都将梯子列表上锁，避免多只猴子同时抢占一个梯子
2. 如有直接共享的数据，则全部都是 imuutable 的
3. 在梯子上向前移动前，会将该梯子上锁以避免猴子重叠

3.6 系统吞吐率和公平性的度量方案

1. 吞吐率：N/T，实现代码如下：

```
public double throughput() {
    return (double) count / secs;
}
```

2. 公平性由以下两个公式计算而得：

$$F(A,B)=\begin{cases}1, & if\ (Y_b-Y_a)*(Z_b-Z_a)\geq 0\\ -1, & otherwise\end{cases}$$

$$F=\frac{\sum_{(A,B)\in\Theta}F(A,B)}{C_N^2},\Theta=\{(A,B)|A\neq B,(B,A)\notin\Theta\}$$

实现代码如下：

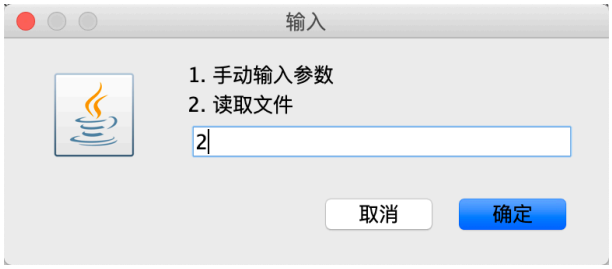
```
public double fairness() {
    int index = 1;
    long cnm = combination(count, 2);
    long timeSum = 0;
    for (Monkey monkey : waitingList) {
        long createTime = monkey.getGroup() * 1000;
        long endTime = monkey.getAliveTime() + createTime;
        Iterator<Monkey> iterator = waitingList.iterator();
        for (int i = 0; i < index; i++) {
            iterator.next();
        }
        while (iterator.hasNext()) {
            Monkey monkey1 = iterator.next();
            long createTime1 = monkey1.getGroup();
            long endTime1 = monkey1.getAliveTime() + createTime1;
            if ((createTime1 - createTime) * (endTime1 - endTime) >= 0) {
                timeSum += 1;
            } else {
                timeSum += -1;
            }
        }
        index++;
    }
    return timeSum / (double) cnm;
}
```

3.7 输出方案设计

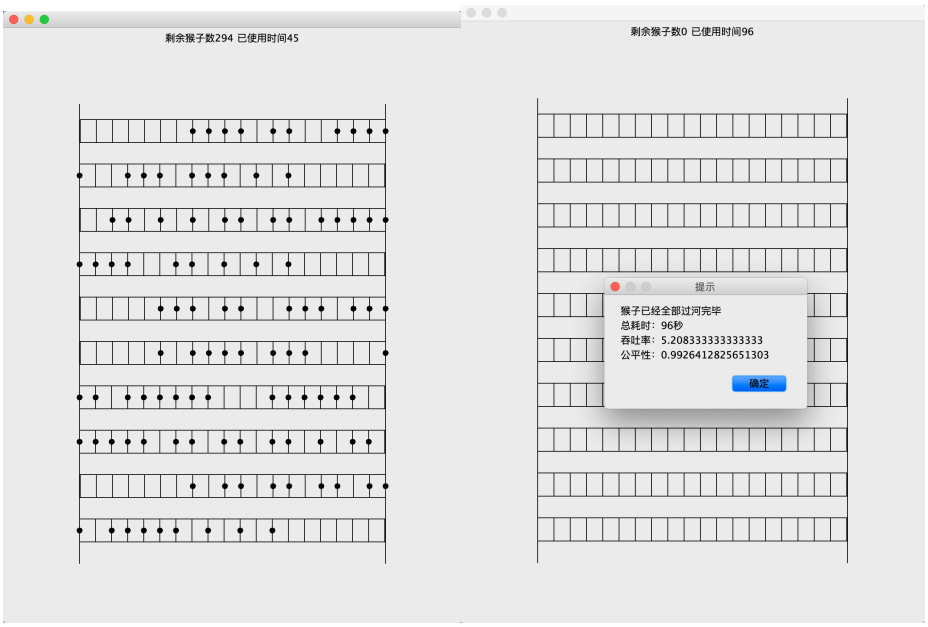
1. 日志使用 log4j，输出样例如图所示：

```
[INFO][2019-06-11 22:54:27,856][abs.Monkey.isOK(Monkey.java:197)]Monkey{id=940, direction='R->L', speed=1}已经出生了 28 秒，到达河对岸，总共耗时 28 秒
[INFO][2019-06-11 22:54:27,856][abs.Monkey.moveAhead(Monkey.java:173)]Monkey{id=960, direction='L->R', speed=1}已经出生了 28 秒，在梯子 abs.Ladder@e57e901 上，方向是L->R，这一秒钟移动到了第 19 根杆上
[INFO][2019-06-11 22:54:27,856][abs.Monkey.isOK(Monkey.java:197)]Monkey{id=942, direction='L->R', speed=2}已经出生了 28 秒，到达河对岸，总共耗时 28 秒
[INFO][2019-06-11 22:54:27,856][abs.Monkey.moveAhead(Monkey.java:173)]Monkey{id=928, direction='R->L', speed=3}已经出生了 28 秒，在梯子 abs.Ladder@6fcbf16c 上，方向是R->L，这一秒钟移动到了第 20 根杆上
[INFO][2019-06-11 22:54:27,856][abs.Monkey.isOK(Monkey.java:197)]Monkey{id=957, direction='R->L', speed=2}已经出生了 28 秒，到达河对岸，总共耗时 28 秒
[INFO][2019-06-11 22:54:27,856][abs.Monkey.isOK(Monkey.java:197)]Monkey{id=936, direction='R->L', speed=2}已经出生了 28 秒，到达河对岸，总共耗时 28 秒
[INFO][2019-06-11 22:54:27,867][abs.Monkey.isOK(Monkey.java:197)]Monkey{id=979, direction='R->L', speed=5}已经出生了 28 秒，到达河对岸，总共耗时 28 秒
[INFO][2019-06-11 22:54:27,867][abs.Monkey.isOK(Monkey.java:197)]Monkey{id=982, direction='L->R', speed=1}已经出生了 28 秒，到达河对岸，总共耗时 28 秒
[INFO][2019-06-11 22:54:27,868][abs.Monkey.moveAhead(Monkey.java:173)]Monkey{id=972, direction='L->R', speed=1}已经出生了 28 秒，在梯子 abs.Ladder@4ab57eaa 上，方向是L->R，这一秒钟移动到了第 20 根杆上
[INFO][2019-06-11 22:54:27,869][abs.Monkey.moveAhead(Monkey.java:173)]Monkey{id=967, direction='R->L', speed=1}已经出生了 28 秒，在梯子 abs.Ladder@5bdd8823 上，方向是R->L，这一秒钟移动到了第 20 根杆上
[INFO][2019-06-11 22:54:27,869][abs.Monkey.moveAhead(Monkey.java:173)]Monkey{id=980, direction='R->L', speed=1}已经出生了 28 秒，在梯子 abs.Ladder@2989cc5d 上，方向是R->L，这一秒钟移动到了第 20 根杆上
[INFO][2019-06-11 22:54:27,886][abs.Monkey.moveAhead(Monkey.java:173)]Monkey{id=970, direction='L->R', speed=1}已经出生了 28 秒，在梯子 abs.Ladder@5406919 上，方向是L->R，这一秒钟移动到了第 20 根杆上
[INFO][2019-06-11 22:54:27,888][abs.Monkey.isOK(Monkey.java:197)]Monkey{id=993, direction='R->L', speed=7}已经出生了 28 秒，到达河对岸，总共耗时 28 秒
[INFO][2019-06-11 22:54:27,888][abs.Monkey.isOK(Monkey.java:197)]Monkey{id=985, direction='R->L', speed=4}已经出生了 28 秒，到达河对岸，总共耗时 28 秒
[INFO][2019-06-11 22:54:27,893][abs.Monkey.moveAhead(Monkey.java:173)]Monkey{id=998, direction='R->L', speed=1}已经出生了 28 秒，在梯子 abs.Ladder@501a21df 上，方向是R->L，这一秒钟移动到了第 18 根杆上
[INFO][2019-06-11 22:54:27,904][abs.Monkey.isOK(Monkey.java:197)]Monkey{id=1000, direction='R->L', speed=5}已经出生了 29 秒，到达河对岸，总共耗时 29 秒
[INFO][2019-06-11 22:54:28,791][abs.Monkey.isOK(Monkey.java:197)]Monkey{id=852, direction='R->L', speed=1}已经出生了 29 秒，到达河对岸，总共耗时 29 秒
```

2. GUI 使用 java.swing 和 java.awt 包下的工具，部分界面截图：



3. 过河可视化见图：



3.8 猴子过河模拟器 v1

3.8.1 参数如何初始化

3.8.2 使用 Strategy 模式为每只猴子选择决策策略

使用 Strategy 设计模式实现，在 Monkey 对象中选择梯子，此时要求传入 LadderSelector 的对象，此部分中使用的是 RandomSelector 的对象，其内部实现仍然是调用已经实现的两个具体的梯子选择方法，如下所示：

```java
/**
 * select ladder.
 * use random number to decide which selection method to use
 * <p>
 * notice: there are two methods:
 * 1. select the ladder which has the same direction as the monkey's
 * 2. select the ladder which is empty
 *
 * @param monkey  which monkey wants to select a ladder?
 * @param ladders ladder list waiting for selection
 * @return the selected ladder object
 */
@Override
public Ladder select(Monkey monkey, List<Ladder> ladders) {
    if (Math.abs(r.nextInt()) % 2 == 0) {
        return accessFirst.select(monkey, ladders);
    } else {
        return empty.select(monkey, ladders);
    }
}
```

3.9 猴子过河模拟器 v2

在不同参数设置和不同“梯子选择”模式下的“吞吐率”和“公平性”实验结果及其对比分析。

3.9.1 对比分析：固定其他参数，选择不同的决策策略

1. n=5, h=20, t=1, N=10, k=3, MV=5 时，使用“选择空梯子”策略：

n	h	t	N	k	MV	总耗时(s)	吞吐率	公平性
5	20	3	10	3	5	18.00	0.56	1.00
5	20	3	10	3	5	21.00	0.48	0.56
5	20	3	10	3	5	30.00	0.33	0.64
平均值	-	-	-	-	-	23.00	0.46	0.73

2. n=5, h=20, t=1, N=10, k=3, MV=5 时，使用“有空位就上梯子”策略：

n	h	t	N	k	MV	总耗时(s)	吞吐率	公平性
5	20	3	10	3	5	23.00	0.43	0.69
5	20	3	10	3	5	27.00	0.37	0.96
5	20	3	10	3	5	20.00	0.50	1.00
平均值	-	-	-	-	-	23.33	0.44	0.88

3.9.2 对比分析：变化某个参数，固定其他参数

1. 使用“选择空梯子”策略，控制 h=20, t=1, N=10, k=3, MV=5，将 n 从 1 取到 5，每个 n 进行三次测试并且计算平均值，制作如下表格：

n	h	t	N	k	MV	总耗时(s)	吞吐率	公平性
n=1	20	3	10	3	5	62.00	0.16	1.00
n=1	20	3	10	3	5	81.00	0.12	0.69

n=1	20	3	10	3	5	63.00	0.16	0.64
n=1						68.67	0.15	0.78
n=2	20	3	10	3	5	63.00	0.16	1.00
n=2	20	3	10	3	5	60.00	0.17	1.00
n=2	20	3	10	3	5	38.00	0.26	0.96
n=2						53.67	0.20	0.99
n=3	20	3	10	3	5	40.00	0.25	0.87
n=3	20	3	10	3	5	45.00	0.22	1.00
n=3	20	3	10	3	5	49.00	0.20	0.91
n=3						44.67	0.23	0.93
n=4	20	3	10	3	5	39.00	0.26	1.00
n=4	20	3	10	3	5	31.00	0.32	0.91
n=4	20	3	10	3	5	25.00	0.40	1.00
n=4						31.67	0.33	0.97
n=5	20	3	10	3	5	18.00	0.56	1.00
n=5	20	3	10	3	5	21.00	0.48	0.56
n=5	20	3	10	3	5	30.00	0.33	0.64
n=5						23.00	0.46	0.73

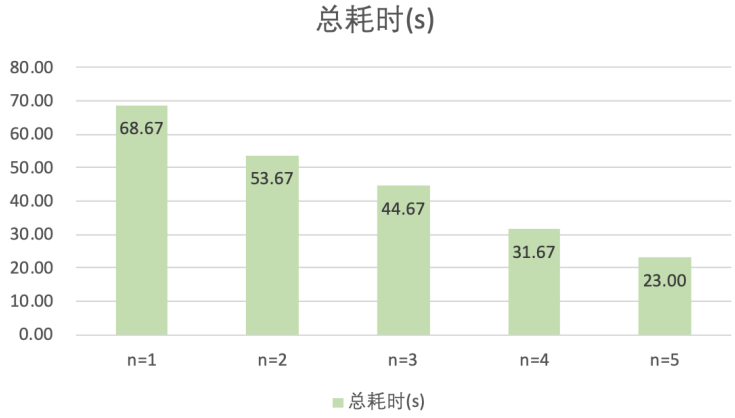
2. 使用“有空位就上梯子”策略，控制 h=20, t=1, N=10, k=3, MV=5，将 n 从 1 取到 5，每个 n 进行三次测试并且计算平均值，制作如下表格：

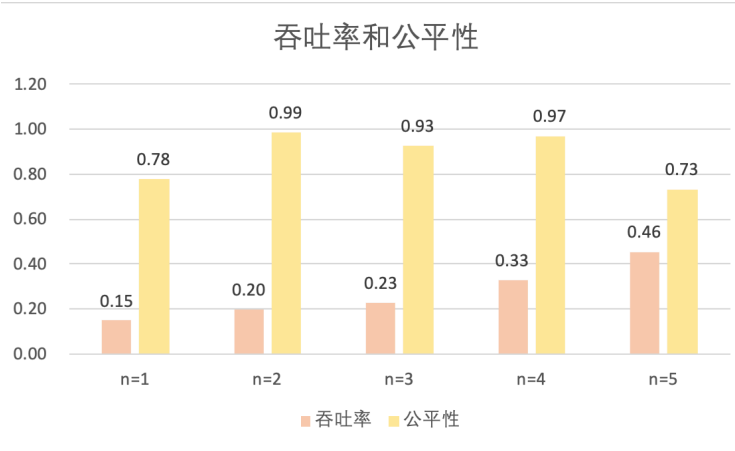
n	h	t	N	k	MV	总耗时(s)	吞吐率	公平性
1	20	3	10	3	5	27.00	0.37	1.00
1	20	3	10	3	5	47.00	0.21	0.78
1	20	3	10	3	5	26.00	0.38	1.00
n=1						33.33	0.32	0.93
2	20	3	10	3	5	29.00	0.34	1.00
2	20	3	10	3	5	32.00	0.31	0.96
2	20	3	10	3	5	31.00	0.32	1.00
n=2						30.67	0.33	0.99
3	20	3	10	3	5	26.00	0.38	0.91
3	20	3	10	3	5	20.00	0.50	0.91
3	20	3	10	3	5	29.00	0.34	0.96
n=3						25.00	0.41	0.93
4	20	3	10	3	5	19.00	0.53	0.91
4	20	3	10	3	5	23.00	0.43	0.96
4	20	3	10	3	5	28.00	0.36	1.00
n=4						23.33	0.44	0.96
5	20	3	10	3	5	23.00	0.43	0.69
5	20	3	10	3	5	27.00	0.37	0.96
5	20	3	10	3	5	20.00	0.50	1.00

n=5						23.33	0.44	0.88

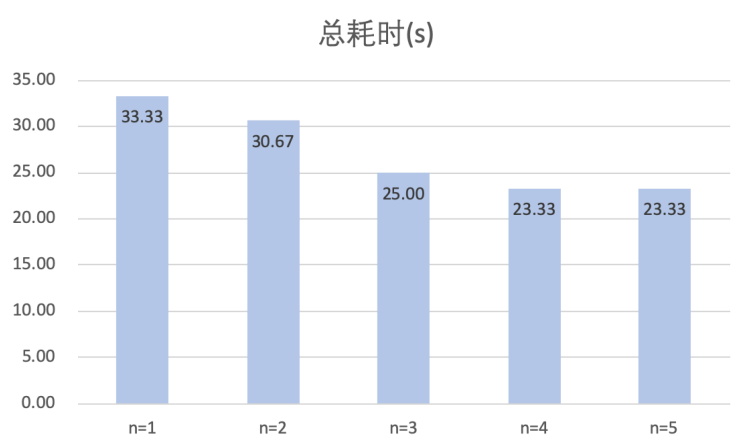
将二者的数据制成柱状图：

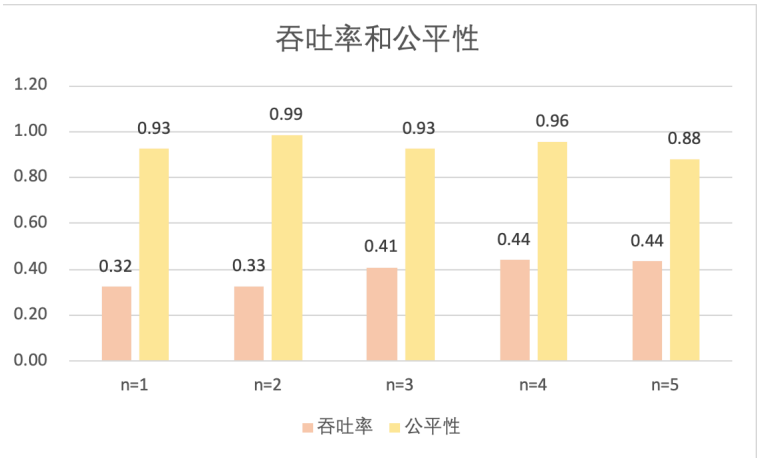
1. “选择空梯子”策略下时，变化 n：





2. “有空位就上梯子”策略下时，变化 n：





3.9.3 分析：吞吐率是否与各参数/决策策略有相关性？

首先，对于我实现的两种梯子选择策略来说，其中的“选择空梯子”无疑是效率极低的一种方式。在对比测试中，当 n=1~4 时，“有空位就上”策略都比“选择空梯子”的吞吐率高了很多，但是在 n=5 时相差无几。

综合对比 n 从 1 到 5 的两种梯子选择方式，分析可知，保证其他参数不变的情况下，梯子数越多，过河速度越快，吞吐率越高，公平性因竞争的关系会存在波动，大致的规律是：囤积的等待过河的猴子数远大于梯子数的时候，会导致公平性的下降。

但是这个对比测试仍有不足之处，一是生成猴子的速度和方向是随机的，所以即使每次参数相同，测试结果也有可能存在很大差距，所以必须要多次实验取平均值减少偶然误差。

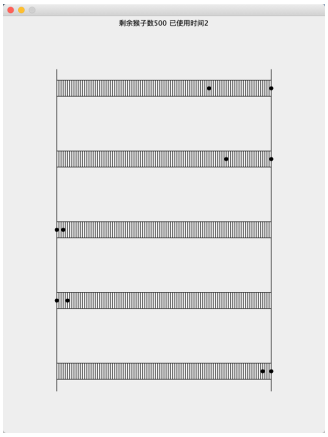
3.9.4 压力测试结果与分析

1. 使用“有空位就上”策略，短时间内产生大量猴子，并且梯子数极少，参数设置为：n=3, h=20, t=1, N=200, k=50, MV=8，测试过程和结果：

2. 使用“有空位就上”策略，短时间内产生大量猴子，并且梯子数极少，参数设置为：n=3, h=20, t=1, N=1000, k=400, MV=8，测试过程和结果：

分析：如果短时间内产生大量的猴子，那么它们之间的竞争会比较激烈，此时带来最直接的影响就是公平性降低。

3. 使用“有空位就上”策略，设置梯子格数较多，并且使猴子速度差异大，参数设置为：n=5, h=100, t=1, N=500, k=20, MV=30，测试过程和结果：



4. 使用“有空位就上”策略，设置梯子格数较多，并且使猴子速度差异大，参数设置为：n=5, h=100, t=1, N=1000, k=20, MV=90，测试过程和结果：

观察过河过程可发现，有的时候某一只猴子速度过慢，会导致其后面的猴子都被它卡住，于是那一把梯子速度极慢，但最后的效果仍然较好。

3.10 猴子过河模拟器 v3

针对教师提供的三个文本文件，分别进行多次模拟，记录模拟结果。

Competition_1.txt		
	吞吐率	公平性
第 1 次模拟	2.2560	0.8926
第 2 次模拟	2.3438	0.9831
第 3 次模拟	2.4390	0.9849
第 4 次模拟	2.5000	0.9825
第 5 次模拟	2.4793	0.9911
第 6 次模拟	2.3622	0.9814
第 7 次模拟	2.3077	0.9119
第 8 次模拟	2.4194	0.9877
第 9 次模拟	2.3077	0.9161
第 10 次模拟	2.4194	0.9938
平均值	2.38345	0.96251
Competition_2.txt		
	吞吐率	公平性
第 1 次模拟	5.4945	0.9795
第 2 次模拟	5.1020	0.9762
第 3 次模拟	4.9505	0.9922
第 4 次模拟	5.4348	0.9851
第 5 次模拟	5.1546	0.9662
第 6 次模拟	4.9505	0.9796
第 7 次模拟	4.9505	0.9850
第 8 次模拟	4.9020	0.9687
第 9 次模拟	5.1021	0.9879
第 10 次模拟	5.00000	0.9574
平均值	5.10415	0.97778
Competition_3.txt		
	吞吐率	公平性
第 1 次模拟	1.2048	0.9980
第 2 次模拟	1.0989	0.9265
第 3 次模拟	1.1628	0.9907
第 4 次模拟	1.1494	0.9556
第 5 次模拟	1.0638	0.9644
第 6 次模拟	1.0989	0.9543

第 7 次模拟	1.0638	0.9438
第 8 次模拟	1.0638	0.9422
第 9 次模拟	1.0989	0.9483
第 10 次模拟	1.0989	0.9479
平均值	1.1104	0.95717

4 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

每次结束编程时，请向该表格中增加一行。不要事后胡乱填写。

不要嫌烦，该表格可帮助你汇总你在每个任务上付出的时间和精力，发现自己不擅长的任务，后续有意识的弥补。

日期	时间段	计划任务	实际完成情况
5.31	9:00-9:30	搭建框架，设计 ADT	完成
	19:30-20:15		
6.1	9:40-12:35	实现猴子产生器和设计梯子选择策略	未完成
	15:00-16:30		
6.2	21:30-22:00		未完成
6.3	14:00-15:30	实现猴子产生器	完成
	19:10-20:30		
	20:30-23:00	设计主程序	未完成
6.4	12:00-12:30		完成
	15:00-15:30	实现一种梯子选择策略	未完成
	20:00-22:20		完成
6.5	9:15-10:40	GUI	未完成
	14:00-16:00	实现第二种梯子选择策略	完成
6.6	15:30-16:00		
	19:30-20:15	GUI	未完成
6.7	9:30-10:00	GUI	未完成
6.8	10:00-11:30	添加新的 ADT	完成
	13:00-15:00	修改 ADT 的部分实现	完成
6.9	9:20-22:00	添加新的 ADT&修改 ADT 的部分实现	完成
6.10	22:00-23:40	GUI	未完成
6.11	一天		完成

6.12	一天	GUI 细节&吞吐率计算&公平性计算	完成
		实验报告	完成

5 实验过程中遇到的困难与解决途径

遇到的难点	解决途径

6 实验过程中收获的经验、教训、感想

6.1 实验过程中收获的经验和教训

6.2 针对以下方面的感受

(1) 多线程程序比单线程程序复杂在哪里？你是否能体验到多线程程序在性能方面的改善？

复杂在所有线程都是一起运行的，如果出现了线程相关的错误，难以复原现场。

(2) 你采用了什么设计决策来保证 threadsafe？如何做到在 threadsafe 和性能之间很好的折中？

直接共享数据时，只共享不可变数据类型，如果需要共享可变数据，需要上锁，为了保证性能，对上锁的部分最小化，一是对锁最小化，二是对上锁的代码最小化

(3) 你在完成本实验过程中是否遇到过线程不安全的情况？你是如何改进的？

遇到了多个猴子同时挤到一个梯子的起点的状况，后来使用同步锁修正。

(4) 关于本实验的工作量、难度、deadline。

适中、适中、时间充裕

(5) 到此为止你对《软件构造》课程的意见和建议。

无

(6) 还有一周就要期末考试了，你准备如何复习？

多看课件，多多总结