

**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 | 罗瑞欣 |
| 学号 | 1170300821 |
| 班号 | 1703008 |
| 电子邮件 | [3102595709@qq.com](mailto:3102595709@qq.com) |
| 手机号码 | 15048260039 |

**目录**

[1 实验目标概述 1](#_Toc1393927)

[2 实验环境配置 1](#_Toc1393928)

[3 实验过程 1](#_Toc1393929)

[3.1 ADT设计方案 1](#_Toc1393930)

[3.2 Monkey线程的run()的执行流程图 1](#_Toc1393931)

[3.3 至少两种“梯子选择”策略的设计与实现方案 2](#_Toc1393932)

[3.3.1 策略1 2](#_Toc1393933)

[3.3.2 策略2 2](#_Toc1393934)

[3.3.3 策略3（可选） 2](#_Toc1393935)

[3.4 “猴子生成器”MonkeyGenerator 2](#_Toc1393936)

[3.5 如何确保threadsafe？ 2](#_Toc1393937)

[3.6 系统吞吐率和公平性的度量方案 2](#_Toc1393938)

[3.7 输出方案设计 2](#_Toc1393939)

[3.8 猴子过河模拟器v1 2](#_Toc1393940)

[3.8.1 参数如何初始化 2](#_Toc1393941)

[3.8.2 使用Strategy模式为每只猴子随机选择决策策略 2](#_Toc1393942)

[3.9 猴子过河模拟器v2 2](#_Toc1393943)

[3.9.1 对比分析：固定其他参数，选择不同的决策策略 3](#_Toc1393944)

[3.9.2 对比分析：变化某个参数，固定其他参数 3](#_Toc1393945)

[3.9.3 分析：吞吐率是否与各参数/决策策略有相关性？ 3](#_Toc1393946)

[3.9.4 压力测试结果与分析 3](#_Toc1393947)

[4 实验进度记录 3](#_Toc1393948)

[5 实验过程中遇到的困难与解决途径 3](#_Toc1393949)

[6 实验过程中收获的经验、教训、感想 4](#_Toc1393950)

[6.1 实验过程中收获的经验和教训 4](#_Toc1393951)

[6.2 针对以下方面的感受 4](#_Toc1393952)

# 实验目标概述

根据实验手册简要撰写。

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

# 实验环境配置

IDEA, jdk9, guava-jdk5

在这里给出你的GitHub Lab5仓库的URL地址（Lab5-学号）。

git@github.com:ComputerScienceHIT/Lab5-1170300821.git

# 实验过程

请仔细对照实验手册，针对三个问题中的每一项任务，在下面各节中记录你的实验过程、阐述你的设计思路和问题求解思路，可辅之以示意图或关键源代码加以说明（但千万不要把你的源代码全部粘贴过来！）。

## ADT设计方案

设计了哪些ADT、各自的作用、属性、方法；

给出每个ADT的specification；

（可选）以类图形式给出多个类之间的关系。

1. Ladder（梯子）

/\*

\* AF(count ladderlist,monkeynumber,direction,start,end,ID)=Ladder;

\* RI: count,ID, monkeynumber must not be negative, direction must not be null.

\* Safety from Representation Exposure: All the key fields are private and immutable.

\*/

主要数据：

|  |  |
| --- | --- |
| public List<Rung> ladderlist | 储存所含的节 |
| private Rung start; | 起始的节 |
| private Rung end; | 终止的节 |
| private String direction; | 梯子上猴子的方向 |

方法：

|  |  |
| --- | --- |
| Getter | Getter |
| Setter | Setter |
| @Override  public boolean equals(Object obj) | Count和ladderlist相等 |
| @Override  public int hashCode() | result = prime \* result + count;  result = prime \* result + ((ladderlist == null) ? 0 : ladderlist.hashCode()); |
| @Override  public String toString() |  |

1. Monkey（猴子）

/\*

\* AF(ID,direction,speed

\* status,localrung,localladder,birthtime,totaltime)=Monkey; RI:direction must

\* be "L -> R" or " R -> L",speed,ID,birthday,totaltime must be positive. Safety

\* From Representation Exposure: all the fields are private, and unmutable.

\* Thread Safety : add lock to localrung.

\*/

implements Runnable

在@Override的run()中实现爬梯子的动作

主要数据：

|  |  |
| --- | --- |
| private int ID = 0; | 猴子的ID |
| private String direction; | 过河的方向 |
| private int speed = 0; | 爬梯子的方向 |
| private int status; | 状态：上梯子前，在梯子上，已过河 |
| private Rung localrung = null; | 所在的梯子的节 |
| private Ladder localladder = null; | 所在的梯子 |
| private long birthday = 0; | 生成的时间 |
| private long totaltime = 0; | 过河的时间 |

方法：

|  |  |
| --- | --- |
| Getter |  |
| Setter |  |
| @Override  public void run() | 选择找梯子的策略，每个猴子开启一个线程，实现找梯子和过河动作。 |
| private void crossRiver() | 在run中调用，完成过河 |
| private Ladder getLadder(FindALadder chooosestrategy, Ladder tmpLadder) | 根据传入的策略寻找梯子 |
| @Override  public String toString() |  |
| @Override  public boolean equals(Object obj) | 判断ID速度和方向是否相等 |
| @Override  public int hashCode() | result=31\*1+ ID;  result=31\*result+((direction==null)?0: direction.hashCode());  result=31\*result+(int)(speed^(speed>>>32)); |

1. Rung（梯子的节）

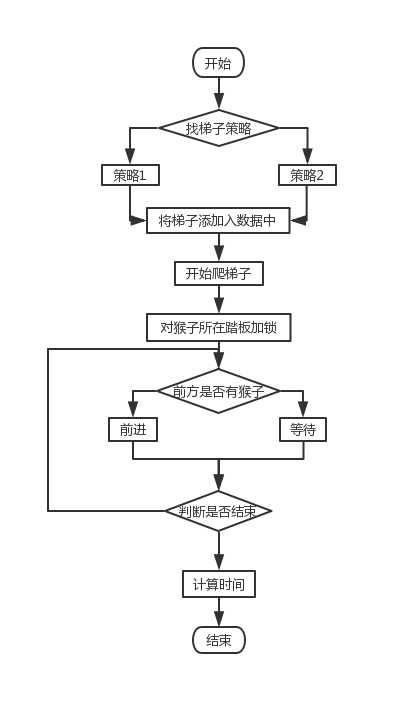
主要数据：

|  |  |
| --- | --- |
| private Monkey monkey = null; | 在该节上的猴子 |
| private int position = 0; | 节在梯子上的位置 |

方法：

|  |  |
| --- | --- |
| Getter |  |
| Setter |  |
| @Override  public boolean equals | 判断数据相等 |
| @Override  public String toString() |  |
| @Override  public int hashCode() | result=prime\*result+((monkey==null)?0: monkey.hashCode());  result=prime \* result + position; |

## Monkey线程的run()的执行流程图



## 至少两种“梯子选择”策略的设计与实现方案

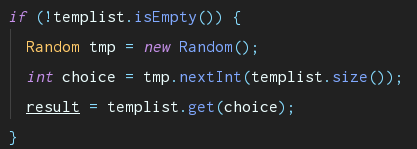
### 策略1

优先选择没有猴子的梯子，若所有梯子上都有猴子，则优先选择没有与我对向而行的猴子的梯子；



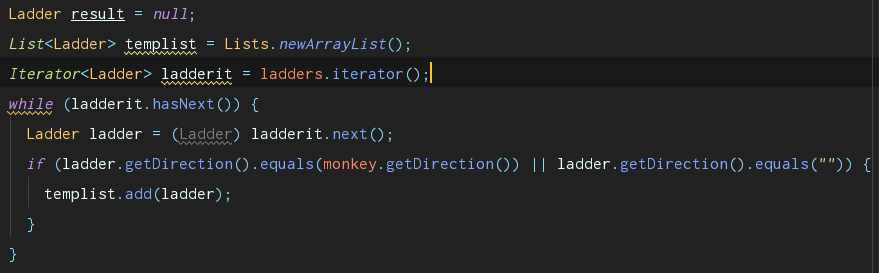
若满足该条件的梯子有很多，则随机选择；



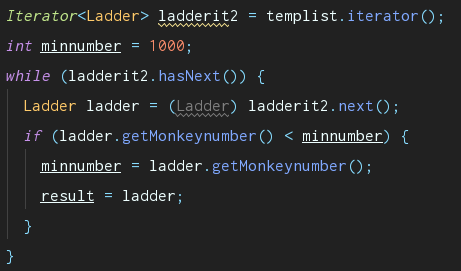


### 策略2

优先选择整体推进速度最快的梯子（没有与我对向而行的猴子、



其上的猴子数量最少、梯子上离我距离最近的猴子的真实行进速度最快）；



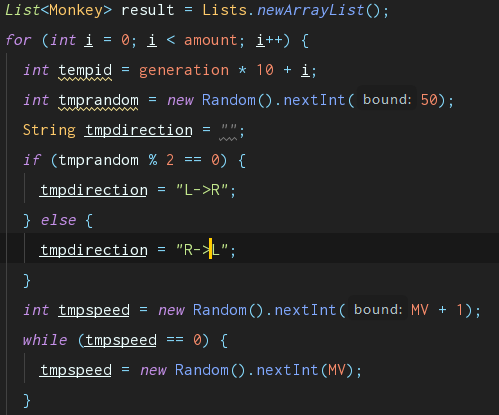
### 策略3（可选）

## “猴子生成器”MonkeyGenerator

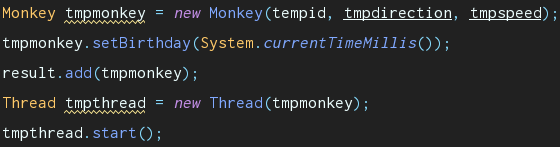
名字ID（int）：按照产生的时间次序进行自然数编号，同一时刻同时生成的猴子的ID 应有不同的编号

方向direction（String）：值随机指定，左岸到右岸(“L->R”)，或者从右岸到左岸(“R->L”)

速度𝑣：正整数，取值范围为[1, 𝑀𝑉]，𝑀𝑉为最大可能的速度。



每个猴子开启一个线程



## 如何确保threadsafe？

加入synchronized关键字，只要将操作共享数据的语句加入synchronized关键字，在某一时段只会让一个线程执行完，在执行过程中，其他线程不能进来执行。方法声明中同步（synchronized）关键字。当它用来修饰一个方法或者一个代码块的时候，能够保证在同一时刻最多只有一个线程执行该段代码。

一、当两个并发线程访问同一个对象object中的这个synchronized(this)同步代码块时，一个时间内只能有一个线程得到执行。另一个线程必须等待当前线程执行完这个代码块以后才能执行该代码块。

二、然而，当一个线程访问object的一个synchronized(this)同步代码块时，另一个线程仍然可以访问该object中的非synchronized(this)同步代码块。

三、尤其关键的是，当一个线程访问object的一个synchronized(this)同步代码块时，其他线程对object中所有其它synchronized(this)同步代码块的访问将被阻塞。

四、当一个线程访问object的一个synchronized(this)同步代码块时，它就获得了这个object的对象锁。结果，其它线程对该object对象所有同步代码部分的访问都被暂时阻塞。

当多个线程要共享一个实例对象的值得时候，那么在考虑安全的多线程并发编程时就要保证3个要素：原子性（Synchronized, Lock）、有序性(Volatile，Synchronized, Lock)、可见性(Volatile，Synchronized,Lock)

当然由于synchronized和Lock保证每个时刻只有一个线程执行同步代码，所以是线程安全的，也可以实现这一功能，但是由于线程是同步执行的，所以会影响效率。

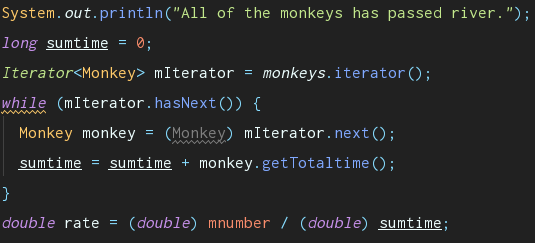
原子性：即一个操作或者多个操作 要么全部执行并且执行的过程不会被任何因素打断，要么就都不执行。在Java中，基本数据类型的变量的读取和赋值操作是原子性操作，即这些操作是不可被中断的，要么执行，要么不执行。

可见性：指当多个线程访问同一个变量时，一个线程修改了这个变量的值，其他线程能够立即看得到修改的值。当一个共享变量被volatile修饰时，它会保证修改的值会立即被更新到主存，当有其他线程需要读取共享变量时，它会去内存中读取新值。普通的共享变量不能保证可见性，因为普通共享变量被修改后，什么时候被写入主存是不确定的，当其他线程去读取时，此时内存中可能还是原来的旧值，因此无法保证可见性。

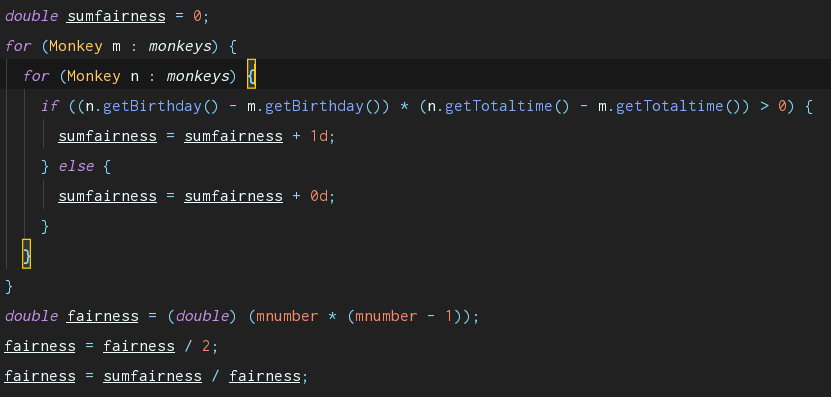
有序性：即程序执行的顺序按照代码的先后顺序执行。在Java内存模型中，允许编译器和处理器对指令进行重排序，但是重排序过程不会影响到单线程程序的执行，却会影响到多线程并发执行的正确性。

## 系统吞吐率和公平性的度量方案

吞吐率：假如 𝑁只猴子过河 的总耗时为 𝑇秒，那么每只猴子的平均耗时 为 𝑋=𝑇𝑁秒 则 吞吐率 𝑇ℎ=𝑁𝑇表征每秒钟可过河的猴子数目 。



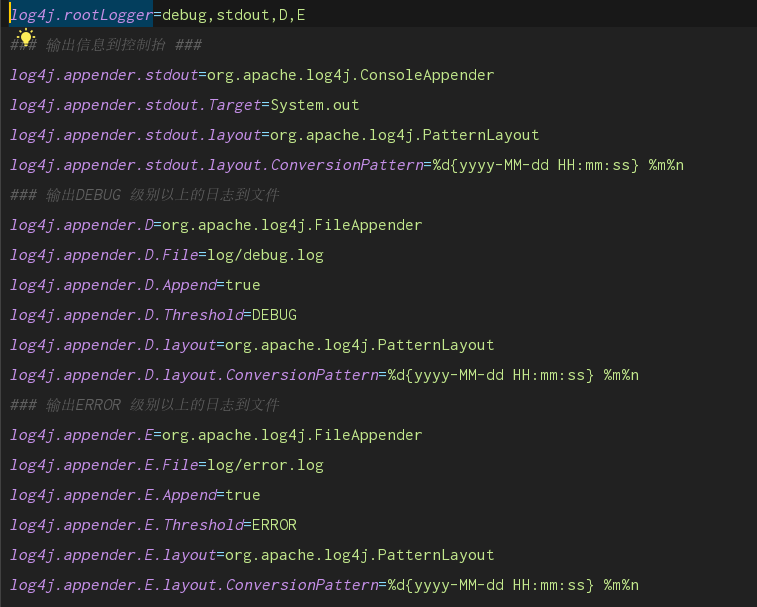
公平性：如果 Monkey 对象A比B出生得早，那么A应该不晚于B抵达对岸，则为公平；若A比B晚到对岸，则为不公平。设A和B的产生时间分别为𝑌𝑎和𝑌𝑏，抵达对岸的时间分别为𝑍𝑎和𝑍𝑏，那么公平性𝐹(𝐴,𝐵)={1,𝑖𝑓 (𝑌𝑏−𝑌𝑎)∗(𝑍𝑏−𝑍𝑎)≥0−1,𝑜𝑡ℎ𝑒𝑟𝑤𝑖𝑠𝑒。对𝑁只猴子两两计算其之间的公平性并综合到一起，得到本次模拟的整体公平性𝐹=Σ𝐹(𝐴,𝐵)(𝐴,𝐵)∈ΘC𝑁2,Θ={(𝐴,𝐵)|𝐴≠𝐵,(𝐵,𝐴)∉Θ}，其取值范围为 [−1,1]。



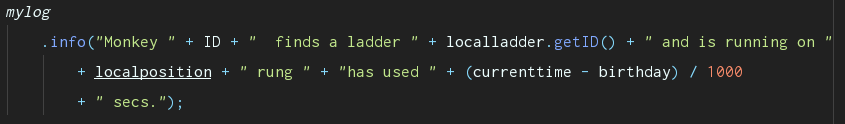
## 输出方案设计

日志：

采用log4j，输入到控制台和文件中，配置如下：



输出的信息如下：







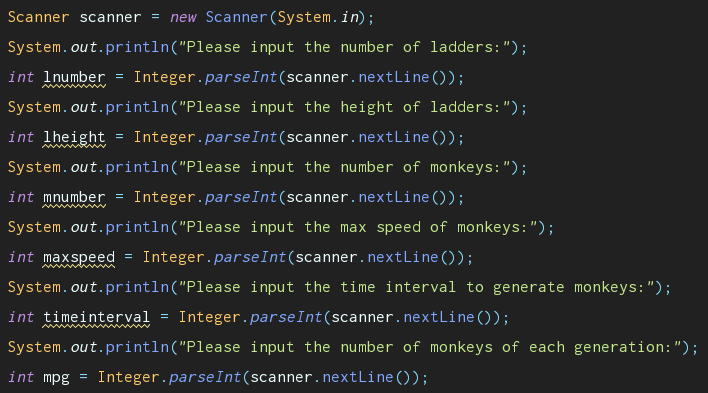
GUI

可视化（可选）

## 猴子过河模拟器v1

### 参数如何初始化

从控制台读入梯子数量、梯子长度、猴子数量、猴子最大速度、产生猴子批次间隔时间和每批猴子数量。

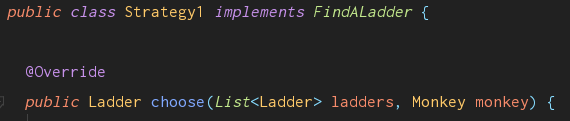


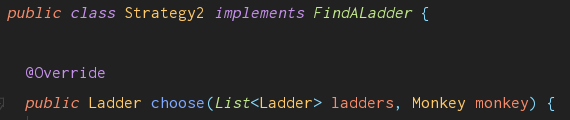
### 使用Strategy模式为每只猴子选择决策策略

设计接口FindALadder



两种具体的策略类继承这个接口，重写choose方法





在类Monkey中的方法run中使用：







## 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。

### 对比分析：固定其他参数，选择不同的决策策略

|  |  |  |
| --- | --- | --- |
|  | 策略1 | 策略2 |
| 吞吐率 | 0.7245 | 0.8974 |
| 公平性 | 0.6291 | 0.8149 |

### 对比分析：变化某个参数，固定其他参数

梯子数量增多，吞吐率有明显提高，公平性略有提高

MonkeySpan增加，吞吐率提高，公平性下降

### 分析：吞吐率是否与各参数/决策策略有相关性？

（1）吞吐率和策略有关，由第一小节知，策略2的吞吐率更高

（2）吞吐率与参数有关，由第二小节知，梯子数目越多，吞吐率越高；MonkeySpan越高，吞吐率越高。

### 压力测试结果与分析

猴子100只，梯子3条，时间间隔1，最大速度10，策略1：

吞吐率：1.579，公平性：0.415

可见猴子数量上升后，在梯子上“堵塞”的情况严重，吞吐率下降严重。

## 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。

|  |  |  |
| --- | --- | --- |
|  | 吞吐率 | 公平性 |
| Competiton\_1.txt |  |  |
| 第1次模拟 | 1.1607 | 0.7483 |
| 第2次模拟 | 0.9723 | 0.6394 |
| 第3次模拟 | 0.9492 | 0.7229 |
| 第4次模拟 | 1.1072 | 0.7039 |
| 第5次模拟 | 1.0703 | 0.6828 |
| 第6次模拟 | 0.9856 | 0.7139 |
| 第7次模拟 | 1.0935 | 0.6249 |
| 第8次模拟 | 0.9583 | 0.8243 |
| 第9次模拟 | 0.9338 | 0.6238 |
| 第10次模拟 | 1.19342 | 0.7239 |
| 平均值 | 1.042432 | 0.70081 |
| Competiton\_2.txt |  |  |
| 第1次模拟 | 2.5122 | 0.4862 |
| 第2次模拟 | 2.5198 | 0.4061 |
| 第3次模拟 | 2.496 | 0.3691 |
| 第4次模拟 | 2.4731 | 0.4071 |
| 第5次模拟 | 2.5071 | 0.3907 |
| 第6次模拟 | 2.5123 | 0.4179 |
| 第7次模拟 | 2.5151 | 0.3582 |
| 第8次模拟 | 2.4197 | 0.4002 |
| 第9次模拟 | 2.5981 | 0.3966 |
| 第10次模拟 | 2.4911 | 0.4014 |
| 平均值 | 2.50445 | 0.40335 |
| Competiton\_3.txt |  |  |
| 第1次模拟 | 0.5032 | 0.4694 |
| 第2次模拟 | 0.523 | 0.4802 |
| 第3次模拟 | 0.4921 | 0.5245 |
| 第4次模拟 | 0.4978 | 0.4928 |
| 第5次模拟 | 0.5279 | 0.4972 |
| 第6次模拟 | 0.5073 | 0.4279 |
| 第7次模拟 | 0.5396 | 0.573 |
| 第8次模拟 | 0.4728 | 0.4729 |
| 第9次模拟 | 0.5307 | 0.5472 |
| 第10次模拟 | 0.5296 | 0.4278 |
| 平均值 | 0.5124 | 0.49129 |

# 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 |
| 6.9 | 下午+晚上 | ADT（除了run）+生成策略 | 完成 |
| 6.10 | 下午+晚上 | Simulator+重写run+v1+v2 | 完成 |
| 6.11 | 下午 | V3+GUI | 完成 |
| 6.12 | 下午 | 测试+报告 | 完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 对于monkey多线程的设计，simulator中的猴子相互协调 | 重新设计monkey中run功能的分配，在run中完成动作，simulator中生成和分析 |
| GUI中图片插入 | Google |
|  |  |

# 实验过程中收获的经验、教训、感想

## 实验过程中收获的经验和教训

## 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？
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？
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？
4. 关于本实验的工作量、难度、deadline。
5. 到此为止你对《软件构造》课程的意见和建议。