哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

Lab Manuals for

Software Construction

Lab-6
Multi-Thread Concurrent Programming

School of Computer Science and Technology

Harbin Institute of Technology

Spring 2018

目录

1. 实验目标

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 和运行性能之间寻求较优的折中，为此计算吞吐率等性能指标，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
- 基本的 GUI 编程

2. 实验环境

实验环境设置请参见 Lab-0 实验指南。

本次实验在 GitHub Classroom 中的 URL 地址为：

https://classroom.github.com/a/aowxhnGf

请访问该 URL，按照提示建立自己的 Lab6 仓库并关联至自己的学号。

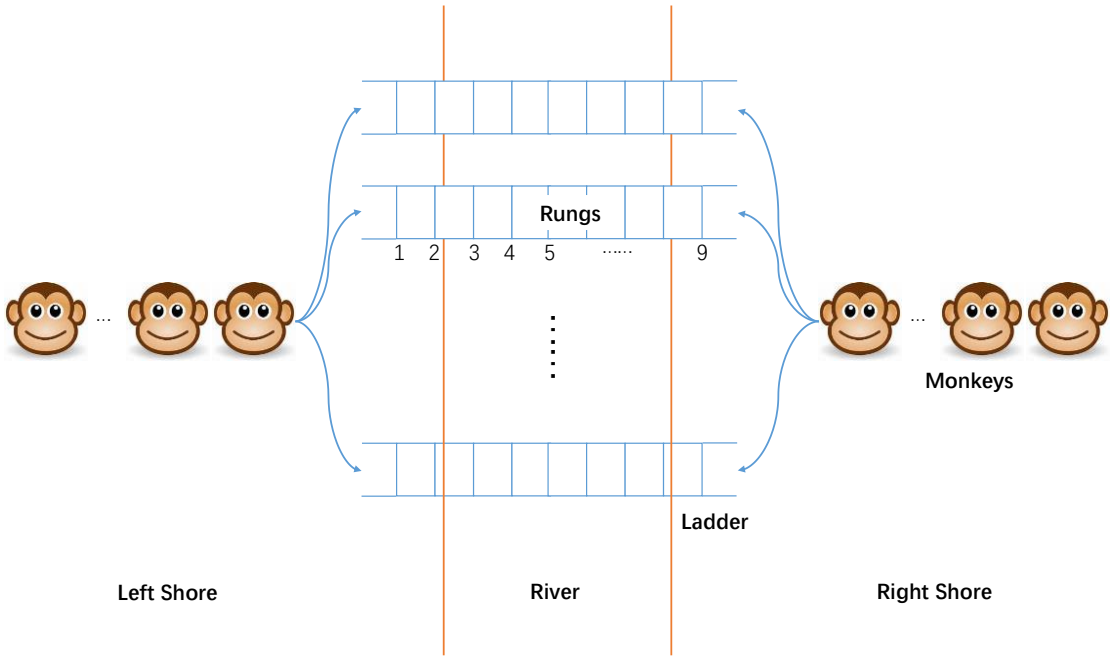
具体目录组织方式参见第 3 节内的说明。请务必遵循目录结构，以便于教师/TA 进行测试。

3. 实验要求

3.1. 需求描述

有一条河，河面上有n架同样的梯子，每个梯子长度为h，意即有h条均匀分布的踏板。

河的左岸有一群猴子，右岸也有一群猴子。左岸的猴子想到右岸，右岸的猴子想到左岸。

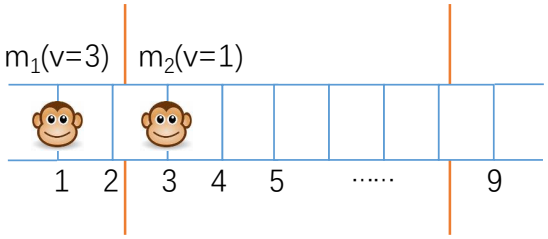


梯子太窄，一只猴子无法越过在其前方同向行进的其他猴子，只能跟随其后（意即：只有在其前方的猴子向前行进腾出了空间，该猴子才能向前进）。猴子无法越过在其前方的对向行进的猴子，也无法在梯子上后退。若在同一架梯子上有两只对向行进的猴子相遇，则此时产生了“死锁”，过河失败。

每个猴子过河的速度不同，其速度v定义为每秒钟可爬过的踏板的数量。在独占一部梯子过河的情况下，一只速度为v的猴子过河所需的时间为$\frac{h}{v}$秒。如果有另一只猴子在它前方但速度较慢，则它的行进速度不得降低。

注：此处一只猴子对另一只猴子的“阻挡”，含义不够清晰，这里用例子解释。

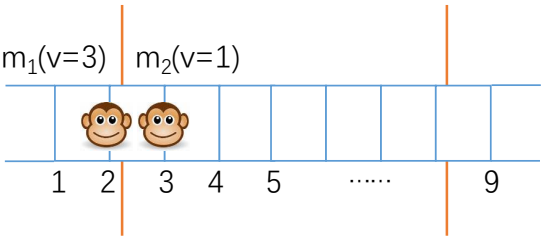
例 1：在某个时刻，猴子m_1位于某架梯子的第 1 个踏板上，其速度为 3，猴子m_2位于同一架梯子的第 3 个踏板上，其速度为 1。假如此时m_1线程在做行动决策，在它独自过河的情况下（理想情况），它应该跳到第$1+3=4$个踏板上，但按照 synchronization/lock 的观点，它需要按次序申请第 2、3、4 三个踏板的“锁”。但是，它观察到自己前方m_2的存在，第 3 个踏板目前由m_2拥有，故m_1无法按预期跳到第 4 个踏板上，它只能降低速度，以速度 1 跳到第 2 个踏板上。



有同学问：m_2也在向前行进，下 1 秒钟m_2应该移动到第 4 个踏板上，所以m_1可以提前做好预判，跳到m_2空出的第 3 个踏板上。——这种情况这违反了后面所提到的不能使用“上帝视角”的原则——猴子只能观察各梯子和各猴子的状

态及其变化，但不能得知其他任何猴子所采取的决策策略。所以，m_1做决策的时候，不能假设自己能够获知m_2的行动策略。

例 2：假如m_1此时在第 2 个踏板上。按照例 1 中的解释，它要申请对第 3、4、5 条踏板的 lock，但第 3 条踏板已被 lock，故在此时m_1的决策只能是“原地不动”。到了下一次做决策的时候，除非m_2已经空出了第 3 条踏板，否则它还是不能行动。



一只猴子在某时刻选择并爬上某个梯子，意味着它从其“出生地”直接跳到了该梯子在猴子所在一侧的第 1 个踏板上。猴子一旦上了某个梯子，就不能在中途跳到别的梯子上。

3.2. 猴子过河模拟器 v1

开发一个模拟猴子过河的仿真程序：

(1) 初始化参数：$n = 1\sim5$，$h = 20$，$t = 1\sim5$，$N = 2\sim100$，$k = 1\sim3$，$MV = 5$，可由开发者在代码中指定具体取值（最好避免！），也可由运行者在命令行或 GUI 以参数的形式指定具体取值，也可由程序从某个外部配置文件中读取参数值（该方法可以支持多次模拟，每次模拟使用不同的参数值配置，均在配置文件里定义，故建议采用）。

(2) 设计 ADT：猴子 `Monkey`、梯子 `Ladder`、以及其他必要的类或接口。

(3) 开发“猴子生成器”`MonkeyGenerator`：每隔t秒钟同时产生k个 `Monkey` 对象（例如：第 0 秒生成k个 `Monkey` 对象，第t秒又同时产生k个 `Monkey` 对象，第 $2t$秒…），并为各只猴子生成以下属性：

- 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 应有区分
- 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。

如果$\frac{N}{k}$不为整数，则最后一次产生的猴子个数为$N\%k$。

(4) 启动过河线程：针对生成的每个 `Monkey` 对象，为其生成一个独立的线程，其目标是通过某个 `Ladder` 对象实现“过河”的目标；

(5) 设计并实现多种梯子选择策略：n个 `Ladder` 对象是在所有猴子的线

程之间共享的数据对象，任何 Monkey 对象被产生出来之后，均可观察到所有 Ladder 对象的当前状态，并根据某种决策策略选择某一架梯子向对岸行进，或者在河岸保持原地观察（暂时不选择具体梯子）。“选择某架梯子”是指：若该梯子靠近该猴子的第 1 个踏板上没有猴子，则猴子跳上该踏板；否则，在原地等待，直到所选梯子的第 1 个踏板空出来才可以跳上去，并且在等待过程中可以切换到其他梯子。猴子在河岸保持原地观察期间，并不需要排队，意即它们对这n个 Ladder 对象的相关踏板产生了“竞争”。选择梯子的时间取决于你的决策算法的执行时间，跳上某梯子的第 1 个踏板不需要耗费时间（即不考虑从猴子出生地到任何梯子的跳跃所需的时间）。

注意：你需要提前设计至少 2 种（或者更多）的决策策略，每个 Monkey 线程在选择梯子时可随机从它们中选择 1 种策略。例如：

- 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、梯子离我最近的猴子的真实行进速度最快）；
- 策略 3：优先选择没有猴子的梯子，若所有梯子上都有猴子，则在岸边等待，直到某个梯子空闲出来；

上述三个策略只是举例说明，并非要求你一定要遵循它们（它们的吞吐率可能不高），故你要仔细思考，设计你自己的决策策略并实现之。设计时，首先要避免死锁，死锁导致T无限大；其次要仔细使用各种 threadsafe 策略，避免滥用而使得并行程序串行化，从而导致耗时增加、吞吐率下降。

使用 Strategy 设计模式实现各 Monkey 对象创建之后选择某种“梯子选择”策略。

注意：设计决策策略时，请不要使用“上帝视角”——猴子只能观察各梯子和各猴子的状态及其变化，但不能得知其他任何猴子所采取的决策策略。

(6) 更新猴子状态并日志：每个猴子过河线程执行时，每隔 1 秒钟更新一次猴子的位置（即猴子每隔 1 秒决定自己的下一步行动），原地不同或根据梯子上的具体情况向前移动若干个横梯，并在 log 日志中记录该 Monkey 对象的当前状态，分为以下三种情况：

- 正在左（右）岸等待，离出生已q秒
- 正在第i架梯子的第j个踏板上，自左向右（自右向左）行进，离出生已q秒

- 已从左（右）岸抵达右（左）岸，共耗时q秒

(7) 线程终止的标准：一旦某个 Monkey 对象已抵达对岸，则其过河成功，线程终止。“抵达对岸”是指：该猴子从其所在梯子的第h条踏板离开，抵达对岸。注意：“猴子已经在某条梯子的第h条踏板上”并不意味着它已经过河，还需下一次决策并行动。当猴子生成器累计产生了N只猴子之后，它停止产生新的猴子，等待所有线程执行结束。

(8) 计算吞吐率和公平性：计算并输出本次仿真的吞吐率和公平性。

- “吞吐率”是指：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$X = \frac{T}{N}$秒，则吞吐率$Th = \frac{N}{T}$表征每秒钟可过河的猴子数目。
- “公平性”是指：如果 Monkey 对象 A 比 Monkey B 出生得更早，那么 A 应该比 B 更早抵达对岸，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Y_a和Y_b，抵达对岸的时间分别为Z_a和Z_b，那么公平性$F(A,B) = \begin{cases} 1, & if\ (Y_b - Y_a) * (Z_b - Z_a) \geq 0 \\ -1, & otherwise \end{cases}$。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$F = \frac{\sum_{(A,B)\in\Theta} F(A,B)}{C_N^2}, \Theta = \{(A,B) | A \neq B, (B,A) \notin \Theta\}$，其取值范围为$[-1,1]$。

你的程序应追求吞吐率尽可能大。公平性并非程序追求的目标，每次模拟时只需计算出公平性的值即可（注：但如果你的程序能在最大化吞吐率的情况下也做到很高的公平性，最好不过了）。

(9) 输出：请在日志和 GUI（可选）输出模拟过河整个过程的各步骤信息，以及本次模拟的公平性和吞吐率。不管是日志还是 GUI 输出，信息要清晰可读、能够清晰看出整个过河过程、本次模拟的各参数设置情况、总体性能情况。（额外计分）如果能使用可视化的形式表现出来，最好不过了！

完成上述任务后，形成版本 1，提交至 Git 仓库。

3.3. 猴子过河模拟器 v2

修改你的版本 1 以形成版本 2。在版本 2 中，所有 Monkey 对象选择梯子的决策策略是相同的。

让你的版本 2 运行多次，各次使用不同的“梯子选择”策略，对比分析不同策略下的“吞吐率”和“公平性”有何差异。（注意：在更换策略时，其他各参数都应保持不变）。进而简要分析你所设计的每种梯子选择策略的适用场合。

请在不同参数设置（$n = 1\sim5$，$h = 20$，$t = 1\sim5$，$N = 2\sim100$，$k = 1\sim3$，$MV = 5$）下进行多次实验，分析你所实现的多种梯子选择策略的“吞吐率”、“公平性”与各参数取值之间是否存在关系。（建议：固定其他参数，只变化某个参数的值，在该参数不同取值情况下对比吞吐率和公平性；然后再更换另一个参数进行变化。例如：让$h = 20$，$t = 3$，$N = 10$，$k = 3$，$MV = 5$，变化n分别为 1、2、3、4、5，进行五次实验，对比五次的性能。）

（可选，额外计分）使用 Lab5 中的 JFreeChart API，生成不同参数下、不同梯子选择策略下吞吐率和公平性的取值对比图，以便于直观展示。

（可选，额外计分）压力测试 1：设计一种参数配置，使得产生的猴子数量非常多、非常密集，而梯子数量有限。观察此时你的程序的吞吐率和公平性表现如何。

（可选，额外计分）压力测试 2：设计一种参数配置，使得各猴子的速度差异非常大。观察此时你的程序的吞吐率和公平性表现如何。

将版本 2 提交至 Git 仓库。

4. 实验报告

针对上述四个编程题目，请遵循 CMS 上 Lab6 页面给出的**报告模板**，撰写简明扼要的实验报告。

实验报告的目的是记录你的实验过程，尤其是遇到的困难与解决的途径。不需要长篇累牍，记录关键要点即可，但需确保报告覆盖了本次实验的所有开发任务。

注意：

- 实验报告不需要包含所有源代码，请根据上述目的有选择的加入关键源代码，作为辅助说明。
- 请确保报告格式清晰、一致，故请遵循目前模板里设置的字体、字号、行间距、缩进；
- 实验报告提交前，请“目录”上右击，然后选择“更新域”，以确保你的目录标题/页码与正文相对应。

5. 提交方式

截止日期：第 16 周周日（2018 年 6 月 17 日）夜间 23:55。截止时间之后通过 Email 等其他渠道提交实验报告和代码，均无效，教师和 TA 不接收，学生本

次实验无资格。

源代码：从本地 Git 仓库推送至个人 GitHub 的 Lab6 仓库内。

实验报告：除了随代码仓库（doc）目录提交至 GitHub 之外，还需手工提交至 CMS 实验 6 页面下。

6. 评分方式

TA 在第 14-15 周实验课上现场验收：学生做完实验之后，向 TA 提出验收申请，TA 根据实验要求考核学生的程序运行结果并打分。现场验收并非必需，由学生主动向 TA 提出申请。

Deadline 之后，教师使用持续集成工具对学生在 GitHub 上的代码进行测试。教师和 TA 阅读实验报告，做出相应评分。