**自助考勤单元测试**

[测试目的: 2](#_Toc2096364895)

[测试数据 : 2](#_Toc1974565303)

[1进入队列算法测试: 3](#_Toc44403017)

[场景1 3](#_Toc176130630)

[场景2 3](#_Toc682429445)

[场景3 3](#_Toc262411665)

[场景4 3](#_Toc884156479)

[场景5 4](#_Toc1147981084)

[场景6 4](#_Toc1542866431)

[2 时间窗口与进入队列算法测试 5](#_Toc375578282)

[测试目的描述 5](#_Toc1511295372)

[测试用例概述: 5](#_Toc399538246)

[测试用例表: 5](#_Toc1892806620)

[测试用例: 6](#_Toc553465581)

[场景1 6](#_Toc285022062)

[场景2 7](#_Toc1224233332)

[场景3 8](#_Toc341329666)

[场景4 9](#_Toc433926017)

[场景5 10](#_Toc1265050088)

[场景6 12](#_Toc1244150879)

测试目的:

自助考勤模块要测试的内容有教师开启自助考勤,教师开启抽点考勤,学生参与自助考勤,学生参与抽点考勤,学生假条提交,教师确认请假,进入队列算法,时间窗口算法,考勤结果算法.

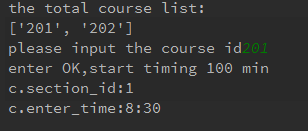
测试数据 :

|  |  |  |
| --- | --- | --- |
| 教师 | 课程号 | 班级 |
| 101 | 201 | 1,2,3 |
| 101 | 202 | 4,5 |
| 102 | 203 | 1,2,3 |
| 102 | 204 | 4,5 |
| 103 | 205 | 1,2,3 |
| 103 | 206 | 4,5 |
| 104 | 207 | 1,2,3,4,5 |

1进入队列测试:

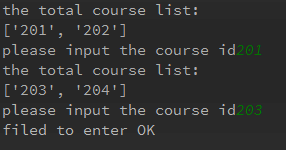
场景1

101 在考勤201,上午8:30,第1节课进入空的队列 (直接进入队列的代码)



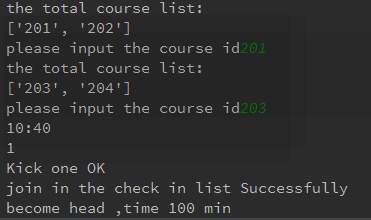
场景2

101正在考勤201,上午8:30,第1节课,102考勤203 (无法进入队列的部分代码)



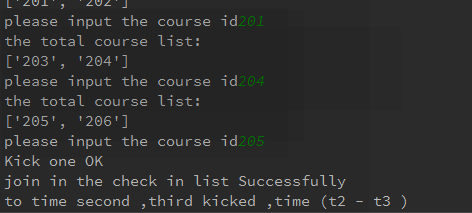
场景3

101 第1节课正在考勤201,第2节课,103考勤203(踢掉队首的代码)



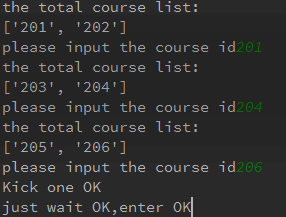
场景4

101第1节课正在考勤201,第1节课,102正在考勤204,第2节课,103要考勤205(第三个人踢掉队首的代码)



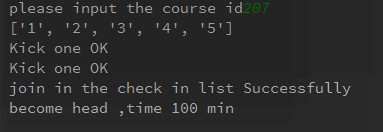
场景5

101第1节课正在考勤201,第1节课,102正在考勤204,第2节课,103要考勤206(第三个人踢掉中间的代码)



场景6

101第1节课正在考勤201;102正在第1节课考勤204;104第二节课要考勤(第三个人踢掉前面两个班,成为队首中间的代码)



经过这6个场景基本覆盖了进入队列的所有代码,为了方便测试,与时间窗口的接口设为pass,场景1: 测试通过,场景2:测试通过,无法进入场景,3:通过,场景4:通过,场景5 :通过,场景6 :通过.

2 时间窗口测试

测试目的

时间窗口只负责计时,时间到了,就结束线程,继续计时下一个.所以重点测试计时结束后能否正常记录下一个.

测试用例概述:

1 单个考勤对象测试 (single 测试)

2 被踢掉,新的计时100秒 (single kick 测试)

3 自然结束,计时下一个 (double 测试)

4 被第三个踢掉,计时下一个 (third kick head 测试)

5 队首计时中,被踢掉,下一个也被踢掉,第三个开始计时(double kick 测试)

6 队首计时中,第二个踢掉,计时第三个(third kick second 测试)

测试用例表:

|  |  |  |  |
| --- | --- | --- | --- |
| 场景名称 | 场景描述 | 预期输出结果 | 测试通过标识 |
| 场景1  (single) | 考勤对象加入队列后自然结束,时间100s | 控制台显示end checkin time 和 cancel time | 与预期结果一样 |
| 场景2  (single kick) | 考勤对象1加入队列20s后,考勤对象2进入队列,踢掉之前的考勤对象,重新计时100s | 考勤对象1开始时间;  20s,考勤对象1被踢出时间;  考勤对象2开始时间;  考勤对象2结束时间; | 考勤对象2的开始结束时间差为100s  考勤对象1被踢出时间等于考勤对象2开始时间. |
| 场景3  (double) | 考勤对象1进入队列,开始计时100s  20s后,考勤对象2进入队列,没有踢掉考勤对象1.考勤对象1结束后,考勤对象2继续计时 | 考勤对象1开始时间;  20s时,考勤对象2开始时间.  考勤对象1 结束时间.  考勤对象2结束时间. | 考勤对象1和2的结束时间时间差为20s;  考勤对象1,2的考勤时间为100s |
| 场景4  (third kick head ) | 9:40,考勤对象1进入队列,计时100s,20s后,考勤对象2进入队列,没有踢掉考勤对象1.30s后,10:30,考勤对象3踢掉考勤对象1,考勤对象2继续计时,考勤对象3等待考勤对象2结束后继续计时,直到考勤结束 | 考勤对象1开始时间;  20s,考勤对象2开始时间;  50s,考勤对象3开始时间;  50s,考勤对象1被踢出时间;  考勤对象2结束时间.  考勤对象3结束时间 | 考勤对象1的考勤时间为50秒;  考勤对象2的考勤时间100s  考勤对象3的考勤时间100s  考勤对象2计时时间100+t2-t3 |
| 场景5  (double kick) | 9:40,考勤对象1进入队列,计时100s,20s后,考勤对象2进入队列,没有踢掉考勤对象1.30s后,10:30,考勤对象3踢掉考勤对象1,考勤对象2,考勤对象3开始计时,直到考勤结束 | 考勤对象1开始时间;  20s,考勤对象2开始时间;  50s,考勤对象3开始时间;  50s,考勤对象1被踢出时间;  50s,考勤对象2被踢出时间  考勤对象3结束时间 | 考勤对象1的考勤时间为50秒;  考勤对象2的考勤时间70s  考勤对象3的考勤时间100s |
| 场景6  (third kick  second) | 9:40,考勤对象1进入队列,计时100s,20s后,考勤对象2进入队列,没有踢掉考勤对象1.30s后,10:30,考勤对象3踢掉考勤对象2,考勤对象1继续计时到结束,考勤对象3开始计时,直到考勤结束 | 考勤对象1开始时间;  20s,考勤对象2开始时间;  50s,考勤对象3开始时间;  50s,考勤对象2被踢出时间;  50s,考勤对象1结束时间  考勤对象3结束时间 | 考勤对象1的考勤时间为100秒;  考勤对象2的考勤时间70s  考勤对象3的考勤时间100s |

测试用例:

场景1

语句:

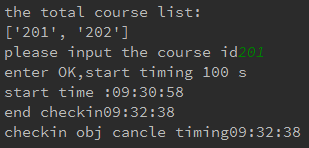
*c = AutoCheckin('w\_101')*

*c.attach(EndcheckinObserver(c))*

*c.attach(TimeWindowObserver(c))*

*c.entry\_list()*

输出结果:



场景2

语句:

*c = AutoCheckin('w\_101')*

*c.enter\_time = '8:30'*

*c.section\_id = 1*

*c.attach(EndcheckinObserver(c))*

*c.attach(TimeWindowObserver(c))*

*c.entry\_list()*

*print 'start time' + time.strftime('%H:%M:%S')*

*time.sleep(20)*

*d = AutoCheckin('w\_102')*

*d.attach(EndcheckinObserver(d))*

*d.attach(TimeWindowObserver(d))*

*d.enter\_time = '10:50'*

*d.entry\_list()*

*print 'start time' + time.strftime('%H:%M:%S')*

输出结果

the total course list:

['201', '202']

please input the course id201

enter OK,start timing 100 s

start time11:36:01

the total course list:

['203', '204']

please input the course id203

Kick one OK

end checkin time :11:36:53\_\_101

cancel time :11:36:53\_\_101

join in the check in list Successfully

become head ,time 100 min

start time11:36:53

end checkin time :11:38:34\_\_102

cancel time :11:38:34\_\_102

场景3

语句:

*c = AutoCheckin('w\_101')*

*c.enter\_time = '9:40'*

*c.entry\_list()*

*c.attach(EndcheckinObserver(c))*

*c.attach(TimeWindowObserver(c))*

*print 'start time' + time.strftime('%H:%M:%S')*

*time.sleep(20)*

*d = AutoCheckin('w\_102')*

*d.enter\_time = '10:00'*

*d.entry\_list()*

*d.attach(EndcheckinObserver(d))*

*d.attach(TimeWindowObserver(d))*

*print 'start time' + time.strftime('%H:%M:%S')*

*time.sleep(30)*

*e = AutoCheckin('w\_103')*

*e.enter\_time = '10:30'*

*e.entry\_list()*

*e.attach(EndcheckinObserver(e))*

*e.attach(TimeWindowObserver(e))*

*print 'start time' + time.strftime('%H:%M:%S')*

输出结果:

the total course list:

['201', '202']

please input the course id201

enter OK,start timing 100 s

start time11:57:43

the total course list:

['203', '204']

please input the course id204

just wait OK,enter OK

start time11:58:07

end checkin time :11:59:23\_\_101

cancel time :11:59:23\_\_101

end checkin time :11:59:43\_\_102

Process finished with exit code 0

场景4

语句:

*c = AutoCheckin('w\_101')*

*c.enter\_time = '9:40'*

*c.entry\_list()*

*c.attach(EndcheckinObserver(c))*

*c.attach(TimeWindowObserver(c))*

*print 'start time' + time.strftime('%H:%M:%S')*

*time.sleep(20)*

*d = AutoCheckin('w\_102')*

*d.enter\_time = '10:00'*

*d.entry\_list()*

*d.attach(EndcheckinObserver(d))*

*d.attach(TimeWindowObserver(d))*

*print 'start time' + time.strftime('%H:%M:%S')*

*time.sleep(30)*

*e = AutoCheckin('w\_104')*

*e.enter\_time = '10:30'*

*e.entry\_list()*

*e.attach(EndcheckinObserver(e))*

*e.attach(TimeWindowObserver(e))*

*print 'start time' + time.strftime('%H:%M:%S')*

输出结果:

the total course list:

['201', '202']

please input the course id201

enter OK,start timing 100 s

start time12:19:40

the total course list:

['203', '204']

please input the course id204

just wait OK,enter OK

start time12:20:03

the total course list:

['205', '206']

please input the course id205

Kick one OK

end checkin time :12:20:37\_\_101

cancel time :12:20:37\_\_101

join in the check in list Successfully

to time second ,third kicked ,time (t2 - t3 )

start time12:20:37

end checkin time :12:21:47\_\_102

end checkin time :12:22:17\_\_103

cancel time :12:22:17\_\_103

分析得:

t1 = 0

t2 = 23

t3 = 57

t1e = 57

t2e = 127

t3e = 157

测试通过.

场景5

语句:

*c = AutoCheckin('w\_101')*

*c.enter\_time = '9:40'*

*c.entry\_list()*

*c.attach(EndcheckinObserver(c))*

*c.attach(TimeWindowObserver(c))*

*print 'start time' + time.strftime('%H:%M:%S')*

*time.sleep(20)*

*d = AutoCheckin('w\_102')*

*d.enter\_time = '10:00'*

*d.entry\_list()*

*d.attach(EndcheckinObserver(d))*

*d.attach(TimeWindowObserver(d))*

*print 'start time' + time.strftime('%H:%M:%S')*

*time.sleep(30)*

*e = AutoCheckin('w\_104')*

*e.enter\_time = '10:30'*

*e.entry\_list()*

*e.attach(EndcheckinObserver(e))*

*e.attach(TimeWindowObserver(e))*

*print 'start time' + time.strftime('%H:%M:%S')*

输出结果:

the total course list:

['201', '202']

please input the course id201

enter OK,start timing 100 s

start time12:33:03

the total course list:

['203', '204']

please input the course id204

just wait OK,enter OK

start time12:33:30

the total course list:

['207']

please input the course id207

Kick one OK

end checkin time :12:34:03\_\_101

cancel time :12:34:03\_\_101

Kick one OK

end checkin time :12:34:03\_\_102

join in the check in list Successfully

become head ,time 100 min

start time12:34:03

end checkin time :12:35:43\_\_104

cancel time :12:35:43\_\_104

分析得:

t1 = 0

t2 = 20

t3 = 50

t1e = 50

t2e = 50

t3e = 100

测试通过.

场景6

语句:

*c = AutoCheckin('w\_101')*

*c.enter\_time = '9:40'*

*c.entry\_list()*

*c.attach(EndcheckinObserver(c))*

*c.attach(TimeWindowObserver(c))*

*print 'start time' + time.strftime('%H:%M:%S')*

*time.sleep(20)*

*d = AutoCheckin('w\_102')*

*d.enter\_time = '10:00'*

*d.entry\_list()*

*d.attach(EndcheckinObserver(d))*

*d.attach(TimeWindowObserver(d))*

*print 'start time' + time.strftime('%H:%M:%S')*

*time.sleep(30)*

*e = AutoCheckin('w\_103')*

*e.enter\_time = '10:30'*

*e.entry\_list()*

*e.attach(EndcheckinObserver(e))*

*e.attach(TimeWindowObserver(e))*

*print 'start time' + time.strftime('%H:%M:%S')*

输出结果:

the total course list:

['201', '202']

please input the course id201

enter OK,start timing 100 s

start time12:45:49

the total course list:

['203', '204']

please input the course id204

just wait OK,enter OK

start time12:46:12

the total course list:

['205', '206']

please input the course id206

Kick one OK

end checkin time :12:46:45\_\_102

just wait OK,enter OK

start time12:46:45

end checkin time :12:47:29\_\_101

cancel time :12:47:29\_\_101

end checkin time :12:48:19\_\_103

分析得:

t1 = 0

t2 = 23

t2e=56

t3 = 56

t1e = 100

t3e = 150

考勤对象1 考勤时间 100

考勤对象2 考勤时间 33

考勤对象3 考勤时间 94

测试通过.

3 开始考勤测试

测试目的

一个考勤对象正常启动依赖的核心是seq id,只要微信号输入正确,seq就要确定下来,否则无法完成下一步的操作.测试教师能否正常使用开始考勤功能,系统能否正确完成seq id 属性detail文件的初始化,sum文件的初始化或更新,seq表的更新,还要完成依赖文件的导入(如果系统路径中没有的话).

测试用例概述

1 教师101打开开始考勤按钮,系统生成detail表,sum表,seq id

2 教师101打开开始考勤按钮,但是seq表缺少上次的seq id

3 教师101打开开始考勤按钮,但是缺少上次的sum表

4 教师101 打开开始考勤按钮,缺少上次的detail表

5 教师101 打开开始考勤按钮,seq id 出错变成了1

测试用例表

|  |  |  |
| --- | --- | --- |
| 用例名称 | 预期输出结果 | 实际结果 |
| 1 | 生成detail文件,更新sum,seq文件 | 与预期结果一致 |
| 2 | 将本次seq\_id + 1,并更新seq文件,生成detail文件,更新sum文件,seq文件 | 与预期结果一致 |
| 3 | 检测出没有sum文件,根据detail文件创建并更新sum文件 | 与预期结果一致 |
| 4 | 根据detail表更新sum文件过程中,如果发现缺少detail文件,仍保留记录,不过sum中那一列置空 | 与预期结果一致 |
| 5 | seq id 显示异常,系统不会直接根据seq表来读取seq id而是判断当前的detail文件最新的序号是否小于seq 表中的序号,如果小于,那么seq为detail文件中最新的序号 | 与预期结果一致 |

4 学生参与自助考勤测试

测试目的

学生能够提交自己的考勤证据,系统接收并得到结果,系统将记录保存到相应的文件中.

测试用例概述

1 学生A正常参与考勤

2 学生A 不在本次考勤的学生列表里面,不能参与考勤

3 学生A和B一起参与考勤

4 学生A提交后,结果为False,再次提交n次,直到考勤成功后,再次参与考勤

5 学生A请假

6 xuesheng A zai shi jian chuangkou zhi wai tijiao

测试用例表

|  |  |  |
| --- | --- | --- |
| 用例名称 | 预期输出结果 | 实际结果 |
| 1 | 增加一次考勤记录考勤结果为空 | 与预期结果一致 |
| 2 | 程序报错 | 与预期结果一致 |
| 3 | 增加两次考勤记录(考勤结果仍为空) | 与预期结果一致 |
| 4 | 增加n次考勤记录(前n-1次为False),没有第n+1条记录 | 与预期结果一致 |
| 5 | 提示学生假条申请一提交,需要上传假条照片 | 与预期结果一致(系统仍会如实判断你是你你在这的结果,不过即使错误不会提示,这个学生一旦请假,系统就不再接受他的考勤需求,除非他取消考勤,那么系统会追加一条考勤结果为空的记录,并提示他再次考勤.考勤结果记为假条提交,(考勤结束时,教师需要手工确认,这里只测试参与考勤,没有考勤结束)) |

5 测试

测试目的

学生能够提交自己的考勤证据,系统接收并得到结果,系统将记录保存到相应的文件中.

测试用例概述

1 学生A正常参与考勤

2 学生A 不在本次考勤的学生列表里面,不能参与考勤

3 学生A和B一起参与考勤

4 学生A提交后,结果为False,再次提交n次,直到考勤成功后,再次参与考勤

5 学生A请假

6 xuesheng A zai shi jian chuangkou zhi wai tijiao

测试用例表

|  |  |  |
| --- | --- | --- |
| 用例名称 | 预期输出结果 | 实际结果 |
| 1 | 增加一次考勤记录考勤结果为空 | 与预期结果一致 |
| 2 | 程序报错 | 与预期结果一致 |
| 3 | 增加两次考勤记录(考勤结果仍为空) | 与预期结果一致 |
| 4 | 增加n次考勤记录(前n-1次为False),没有第n+1条记录 | 与预期结果一致 |
| 5 | 提示学生假条申请一提交,需要上传假条照片 | 与预期结果一致(系统仍会如实判断你是你你在这的结果,不过即使错误不会提示,这个学生一旦请假,系统就不再接受他的考勤需求,除非他取消考勤,那么系统会追加一条考勤结果为空的记录,并提示他再次考勤.考勤结果记为假条提交,(考勤结束时,教师需要手工确认,这里只测试参与考勤,没有考勤结束)) |