48024 Applications Programming

Assignment 1

Topics: OO design, standard patterns, lists
Objectives: This assignment supports objectives 1 - 3
Due date: 11:59pm Monday 19th of September 2016
Weight: 30%

1. Individual work

All work is individual. You may discuss ideas, approaches and problems, but you should write every line of code yourself except for code copied from the lecture notes, lecture code or lab code. More information about Academic Misconduct can be found at: http://www.gsu.uts.edu.au/rules/student/section-16.html

2. Specification

A secure research facility has hired you to develop a new system for the lifts of its secure building. The system needs to know where everyone in the building is at every moment. To use the lifts, a person must be added to the system and given an ID card. This ID card must be swiped when making a trip on the lift so that the lift can track the movement of that person.

The building has 6 floors and 3 lifts. Each lift services a different range of levels: Lift 1 services levels 1-6, Lift 2 services levels 2-6 and Lift 3 services levels 2-5. The building entrance is on level 2 and the initial resting positions of all 3 lifts is level 2.

When a person registers for an ID card, the system records their initial level as 2. A person can also be removed from the system, but only if they are on level 2 and not currently waiting for or aboard a lift.

When calling a lift, a person swipes his/her ID card, selects a destination level/floor, and then waits. The system uses an algorithm to select the most suitable lift and reports the number of that lift to the caller. When the lift reaches the caller's current level, the caller boards the lift unless the lift is heading in the opposite direction to that of the caller's destination. When the lift reaches the caller's destination, the caller alights from the lift. The lift will stop at a level if any person wants to board or alight.

The most suitable lift for a caller is determined by a suitability function. If a lift's service range doesn't include the caller's current level or destination level, that lift's suitability is 0. Otherwise if a lift is moving away from the caller, its suitability is 1. Otherwise if a lift is moving in the opposite direction that the caller wants to go in, then its suitability is N + 1 - D where N is the highest minus the lowest level of the building and D is the distance in levels between the lift and the caller. Otherwise, the lift's suitability is N + 2 - D. The most suitable lift for a caller is the lift with the highest suitability.

An aside...

While reading the the first part of the specification, you will notice there is a lot going on.

- *How many functions did you identify?*
- *How many classes did you identify?*
- *What are the fields in each class?*
- *How many goals did you identify?*
- *How many patterns did you think of that might be applicable?*

This assignment will be challenging and you will probably want to manage your time well.

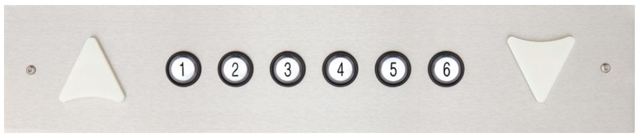
- *How long do you think it will take you to code the functions?*
- *How long do you think it will take you to code each goal?*

A good rule of thumb is to think of an estimate, and then multiply that number by 3 or 4!
To manage your time well, you may need to figure out which parts of the assignment you can start early.

- *Which parts can you start now?*
- *Which parts can you start in week 5?*

If you complete parts in the same week that you learn the topics (while they are fresh in your mind), they will take less time to complete.

The User Interface



Below is a sample I/O trace. Not every conceivable scenario is shown below and you should submit your code to PLATE to see what specific scenarios are tested.

Basic usage

When the program launches, you will see a mode prompt which can be used to change the behaviour of the program.

```
Mode:
```

For the moment, let's leave this blank and press enter. The menu is now shown. Inputting an unrecognised command results in the help being shown:

```
Choice (a/r/p/c/l/o/x): ?
```

```
Menu
a = add person
r = remove person
p = show people
c = call lift
l = show lifts
o = operate
```

The user can add, remove and show people registered in the system with the ‘a’, ‘r’ and ‘p’ options. Each person must have a unique ID:

```
Choice (a/r/p/c/l/o/x): a
Person ID: 1
Name: Sam Worthington
Choice (a/r/p/c/l/o/x): a
Person ID: 1
ID already exists
Choice (a/r/p/c/l/o/x): a
Person ID: 2
Name: Nicole Kidman
Choice (a/r/p/c/l/o/x): a
Person ID: 3
Name: Hugh Jackman
Choice (a/r/p/c/l/o/x): p
Sam Worthington(1) on level 2
Nicole Kidman(2) on level 2
Hugh Jackman(3) on level 2
Choice (a/r/p/c/l/o/x): r
Person ID: 1
Choice (a/r/p/c/l/o/x): r
Person ID: 1
No such person
Choice (a/r/p/c/l/o/x): p
Nicole Kidman(2) on level 2
Hugh Jackman(3) on level 2
```

The lifts are shown with the ‘l’ option:

```
Choice (a/r/p/c/l/o/x): l
Lift 1  |-0----|
Lift 2   |0----|
Lift 3   |0---|
```

Each lift is displayed as a string of dashes with the left side representing the bottom of the lift and the right side representing the top of the lift. One of the dashes is replaced by a number indicating how many passengers are currently aboard that lift. The position of this number also indicates the location of the lift within the shaft. Notice that Lift 2 and Lift 3 are shifted right by one level because those lifts start at level 2. Your code must not have any special cases to deal with different lifts. For example, you cannot write code like this: if (liftNumber != 1) { add an extra space }.

A lift can be called with the ‘c’ option:

```
Choice (a/r/p/c/l/o/x): c
Person ID: 1
No such ID
Choice (a/r/p/c/l/o/x): c
Person ID: 2
Destination level: 4
Choice (a/r/p/c/l/o/x): p
Nicole Kidman(2) on level 2 waiting to go to level 4
Hugh Jackman(3) on level 2
Choice (a/r/p/c/l/o/x): l
Lift 1  |-0----|
Lift 2   |0----|
Lift 3   |0---|
```

The most suitable lift is chosen and the caller is added to the lift’s queue. In this case, all 3 lifts are equally suitable so the first lift will be chosen. If there was no suitable left (e.g. the destination was not serviceable by any lift), the call is discarded.

The lifts are operated via the ‘o’ option. Here, each lift use this algorithm:

- If the lift is currently stationary, check if there is anyone in the queue to be picked up or if there is any passenger to be dropped off, and set the direction of the lift toward the pickup or dropoff point.
- Before moving, let any passengers who want to alight here alight (they are then removed from the passengers list), and let any people in the queue board if they’re heading in the same direction as the lift and they’re on the same level that the lift is currently on (they are then removed from the queue).
- If the lift is now empty, the lift becomes idle (set the direction to 0).
- Otherwise, move the lift and all its passengers one level in the current direction. The lift stops if it reaches the top or bottom (set the direction to 0).

Now that Nicole has made a call for a lift, the operate option will work as follows:

```
Choice (a/r/p/c/l/o/x): o
Choice (a/r/p/c/l/o/x): p
Nicole Kidman(2) on level 2 going to level 4
Hugh Jackman(3) on level 2
Choice (a/r/p/c/l/o/x): l
Lift 1  |-1----| UP
Lift 2   |0----|
Lift 3   |0---|
```

Nicole has boarded lift 1 and is now going UP to level 4, but the lift hasn’t moved yet. Pressing ‘o’ again:

```
Choice (a/r/p/c/l/o/x): o
Choice (a/r/p/c/l/o/x): p
```

```
Nicole Kidman(2) on level 3 going to level 4
Hugh Jackman(3) on level 2
Choice (a/r/p/c/l/o/x): l
Lift 1  |--1---| UP
Lift 2   |0----|
Lift 3   |0---|
```

The lift has moved up to level 3, and Nicole has moved along with the lift. Again:

```
Choice (a/r/p/c/l/o/x): o
Choice (a/r/p/c/l/o/x): p
Nicole Kidman(2) on level 4 going to level 4
Hugh Jackman(3) on level 2
Choice (a/r/p/c/l/o/x): l
Lift 1  |---1--| UP
Lift 2   |0----|
Lift 3   |0---|
```

The lift with Nicole have moved up to level 4 and has stopped, but Nicole hasn't alighted yet. Again:

```
Choice (a/r/p/c/l/o/x): o
Choice (a/r/p/c/l/o/x): p
Nicole Kidman(2) on level 4
Hugh Jackman(3) on level 2
Choice (a/r/p/c/l/o/x): l
Lift 1  |---0--|
Lift 2   |0----|
Lift 3   |0---|
```

Nicole has now alighted and there are zero passengers aboard.

The 'x' option exits the program.

```
Choice (a/r/p/c/l/o/x): x
```

Modes

When starting the program, the mode prompt allows for the input of a string of special mode codes in any order.

- 'w' means show how many people are waiting for a lift on each level of the building.
- 'i' means show how many people are idle on each level (i.e. not aboard a lift and not waiting for a lift).
- 'a' means the lifts will be automatically be displayed along with the program's main menu.

For example, if you input "wi" or "iw" as the mode, then the waiting and idle numbers will be displayed for each level:

```
Idle:    1   2
Waiting:   1
Lift 1  |-0----|
Lift 2   |0----|
Lift 3   |0---|
```

This indicates that 1 person is idle on level 1, 2 people are idle on level 5 and 1 person is waiting for a lift on level 3. If the mode is “w”, then just the waiting numbers will be displayed:

```
Waiting:   1
Lift 1  |-0----|
Lift 2   |0----|
Lift 3   |0---|
```

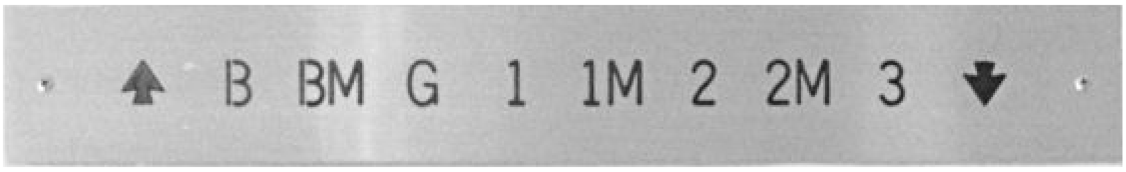
Let’s run the program with all mode options enabled:

```
Mode: iwa
Idle:
Waiting:
Lift 1  |-0----|
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x):
```

As you can see, the lifts are automatically shown along with the menu. With the ‘a’ mode turned on, there is no longer any need to use the ‘l’ option to view the lifts. Also, with the ‘w’ and ‘i’ modes turned on, there is little reason to use the ‘p’ option to view the people. This allows us to run a scenario just with a combination of the call and operate menu options (after first adding some people):

```
Choice (a/r/p/c/l/o/x): a
Person ID: 1
Name: Bridget Jones
Idle:     1
Waiting:
Lift 1  |-0----|
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): a
Person ID: 2
Name: Mark Darcy
Idle:     2
Waiting:
Lift 1  |-0----|
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): c
```

```
Person ID: 1
Destination level: 3
Idle:     1
Waiting:  1
Lift 1  |-0----|
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): c
Person ID: 2
Destination level: 6
Idle:
Waiting:  2
Lift 1  |-0----|
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:
Waiting:
Lift 1  |-2----| UP
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:
Waiting:
Lift 1  |--2---| UP
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:      1
Waiting:
Lift 1  |--1---| UP
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:      1
Waiting:
Lift 1  |---1--| UP
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:      1
Waiting:
Lift 1  |----1-| UP
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:      1
Waiting:
Lift 1  |-----1|
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:      1  1
Waiting:
Lift 1  |-----0|
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): c
Person ID: 1
Destination level: 6
Idle:         1
Waiting:   1
Lift 1  |-----0|
Lift 2   |0----|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:         1
Waiting:   1
Lift 1  |-----0|
Lift 2   |-0---| UP
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:         1
Waiting:
Lift 1  |-----0|
Lift 2   |-1---| UP
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:         1
Waiting:
Lift 1  |-----0|
Lift 2   |--1--| UP
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:         1
Waiting:
Lift 1  |-----0|
Lift 2   |---1-| UP
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:         1
Waiting:
Lift 1  |-----0|
Lift 2   |----1|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): o
Idle:         2
Waiting:
Lift 1  |-----0|
Lift 2   |----0|
Lift 3   |0---|
Choice (a/r/p/c/l/o/x): x
```



Requirements

- Your design must consist of exactly the following set of classes and fields. All fields must be declared private. You may not add or remove classes or fields. You may add constructors, functions and procedures. (Copy and paste this into a new project)

```
public class Building {
    private int entrance = 2;
    private int bottom = 1;
    private int top = 6;
    private LinkedList<Lift> lifts = new LinkedList<Lift>();
    private LinkedList<Person> people = new LinkedList<Person>();
    private String mode;
}
public class Lift {
    private int number;
    private int bottom;
    private int top;
    private int level;
    private int direction;
    private LinkedList<Person> passengers = new
LinkedList<Person>();
    private LinkedList<Person> queue = new LinkedList<Person>();
}
public class Person {
    private int id;
    private String name;
    private int level;
    private int destination;
    private boolean aboard;
}
public class In {
    private static final Scanner scanner = new Scanner(System.in);
}
```

- Constructor requirements:
    - The Person constructor must initialise ALL 5 fields from parameters.
    - The Lift constructor must initialise the number, bottom, top and level fields from parameters. The direction always starts at 0.

    - The Building constructor must initialise the mode field by reading it from the user, and must add the 3 lifts to the list of lifts.
    - Constructor parameters must be listed in the same order as the fields.
- Field requirements:
    - The lift direction is always either -1 (down), 0 (stationary) or 1 (up).
    - The lift queue contains the list of people waiting for that particular lift.
    - The lift passengers list contains the people currently inside that particular lift.
- toString() function requirements:
    - The Person toString() function must return a string in one of these forms depending on whether the person is (1) currently on board a lift, (2) waiting for a lift, or (3) neither on board a lift nor waiting for a lift:
        - Ryan Heise (1) on level 2, going to level 4
        - Ryan Heise (1) on level 2, waiting to go to level 4
        - Ryan Heise (1) on level 2
          where “Ryan Heise” and 1 indicate the person’s name and ID.
    - The Lift toString() function must return a string beginning with the word “Lift”, then a space, then the lift number, then two spaces. If the bottom of the lift is at level 1, then the next part of the string will be “|” followed by a string of dashes “---...etc” followed by another “|”. The number of dashes is determined by how many levels are in the lift’s range, and one of the dashes should be replaced by a number indicating how many passengers are currently aboard that lift. The particular dash that gets replaced by this number depends on the current location of the lift. If the lift is currently at its bottom, then replace the leftmost dash. If the lift is currently at the top, replace the rightmost dash. If the lift is somewhere in the middle, replace the dash at the corresponding position. After the final “|”, the string may contain a space followed by the word “UP” or “DOWN” depending on whether the lift is currently moving up or down. This part of the string is absent if the lift is stationary. If the bottom of the lift is N levels higher than 1, then it should be shifted right N spaces so that it all are aligned at the corresponding levels. Examples:
        - `Lift 1  |-2----| UP`
        - `Lift 2   |---0-|`
        - `Lift 3   |---1| DOWN`
- Your main method must be defined in the Building class.
- You must not use hard coded literals. i.e. Don’t write the numbers 1, 6 or 2 anywhere in your code. Always refer to the bottom, top and entrance variables.
- You must submit your program to PLATE and match PLATE’s output to receive marks.

Beyond this, you are free to make your own design decisions, however, a design guide will be posted to PLATE in week 5 which will offer strong recommendations on how you should design and build your solution for best results.

Take a short break, and then continue reading. What follows is important!

4. Expected workload

The time to do the assignment to a credit/distinction level has been estimated at 25 hours for a student of average ability who has completed all the tutorial and lab exercises.

5. Online support

The Assignment 1 discussion board has been set up so that students can ask questions, and other students can reply. I will post a reply only if I think the student response was wrong, or in the case of correcting a mistake in the assignment specification.

You must not post Java code to the discussion board. The board is there to help you, not to provide the solution. Posting your code is **academic misconduct** and will reported. Each time this rule is violated, I will delete the code and post a comment of the form: “Strike 1: Posting code”. After 3 strikes, the discussion board will be deleted because it did not work.

FAQs (Frequently Asked Questions) and their answers are posted on UTSOnline in Assignments/1/faq. If you have a question, check the FAQ first; it may already be answered there. You should read the FAQ at least once before you hand in your solution, but to be safe check it every couple of days. Anything posted on the FAQ is considered to be part of the assignment specification. The FAQ will be frozen (no new entries) two days before the due date; no questions will be answered after it is frozen.

If anything about the specification is unclear or inconsistent, contact me and I will try to make it clearer by replying to you directly and posting the common questions and answers to the FAQ. This is similar to working on the job, where you ask your client if you are unsure what has to be done, but then you write all the code to do the task. Email Ryan.Heise@uts.edu.au to ask for any clarifications or corrections to the assignment.

6. PLATE marking

Your solution is marked for correctness by PLATE (https://plate.it.uts.edu.au/) by comparing the output of your system to the output of the benchmark system. You can submit a solution to PLATE many times; I urge you to do this, so you receive credit for your work.

PLATE will test the features of your program in a certain order: Classes and fields, then constructors, then goals from basic to advanced. PLATE cannot test the more advanced goals until the basic goals are working. For example, you must implement “call” before “add person”, because it is impossible for a person to call a lift if no people have been added yet. To receive marks, you must pass PLATE’s test cases in the order in which PLATE tests them.

Your code is marked by software, so you can get a good mark by fooling or spoofing the software. If you spoof a task worth N marks, you receive a penalty of 2*N marks.

7. Submission and return

Your solution is to be submitted to PLATE at https://plate.it.uts.edu.au/ under Applications Programming / Assessments / Assignment 1. Your provisional mark and feedback is generated immediately each time you submit to PLATE. However, analysis of spoofing, plagiarism and collusion is done in the two weeks following the due date. If you are suspected of Academic Misconduct, I will forward your case to the Misconduct Committee and will notify you by email.

There is no scheduled late submission period. An extension of up to one week may be given by the subject coordinator before the due date; you have to supply documentary evidence of your claim. An extension CANNOT be given after the due date.

You may also apply for special consideration for reasons including unexpected health, family or work problems. More information about how to apply for special consideration can be found at http://www.sau.uts.edu.au/assessment/consideration.html.

8. Marking scheme

Task	**Mark**
Class and field declarations	5%
Constructors	5%
toString functions	10%
Menu help	5%
Add person	5%
Show people	3%
Remove person	5%
Show lifts	2%
Call lift 1	10%
Operate lift 1	15%
Modes	13%
Suitability function	10%
Call and operation of multiple lifts	12%