Next Level Abstraction

Istvan Nagy

Introduction

- Lead Software Engineer at EPAM
- 12+ years of Java experience
- Member of the EPAM Debrecen Java Community

- The one who loves giraffes
- The rebel (as illustrated)
- The punisher

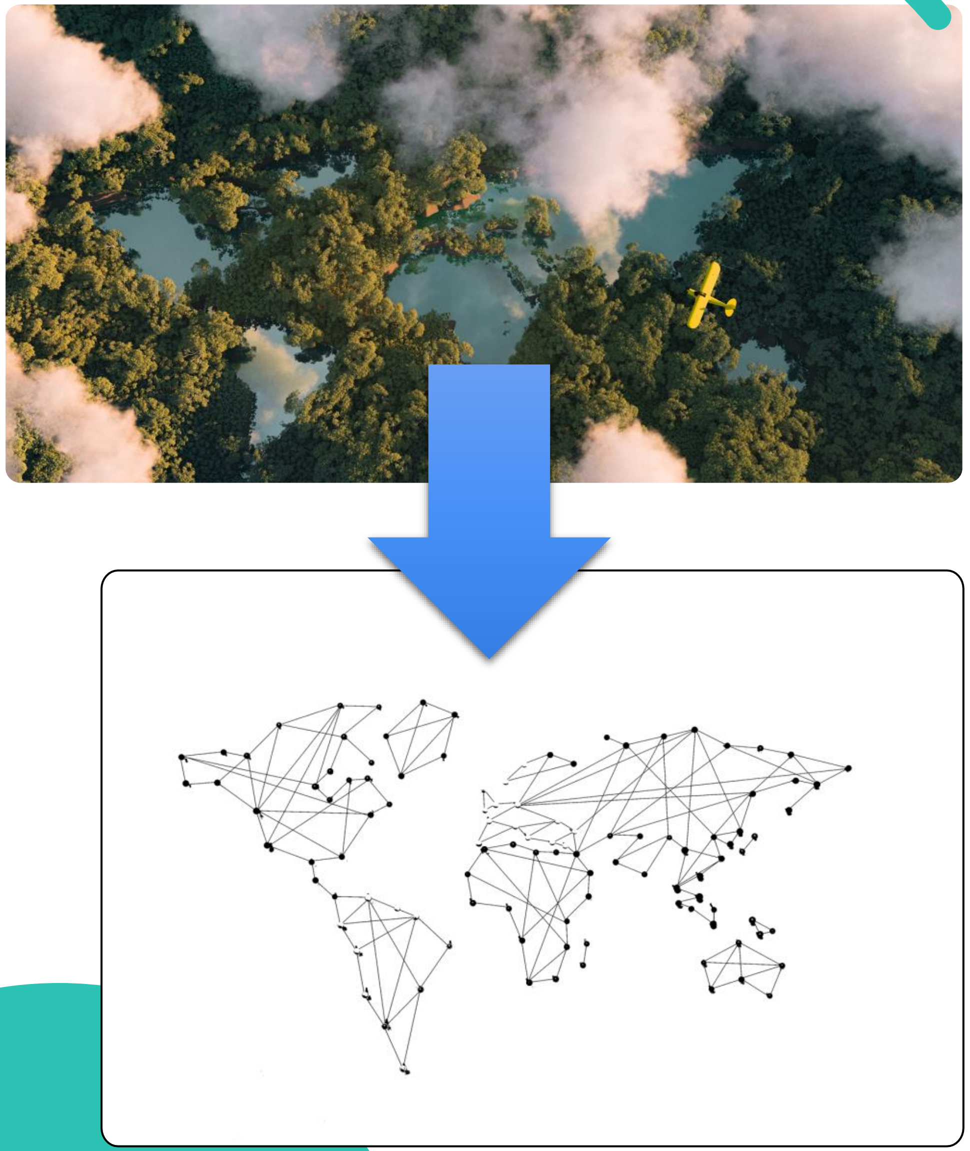
Plan “A”

This is <u>**not**</u> a presentation. It should be a dialogue

The World

as we know it

How do we make things work with the computer?



They aren't! We need to simplify a lot, in order to make it work.
But why?

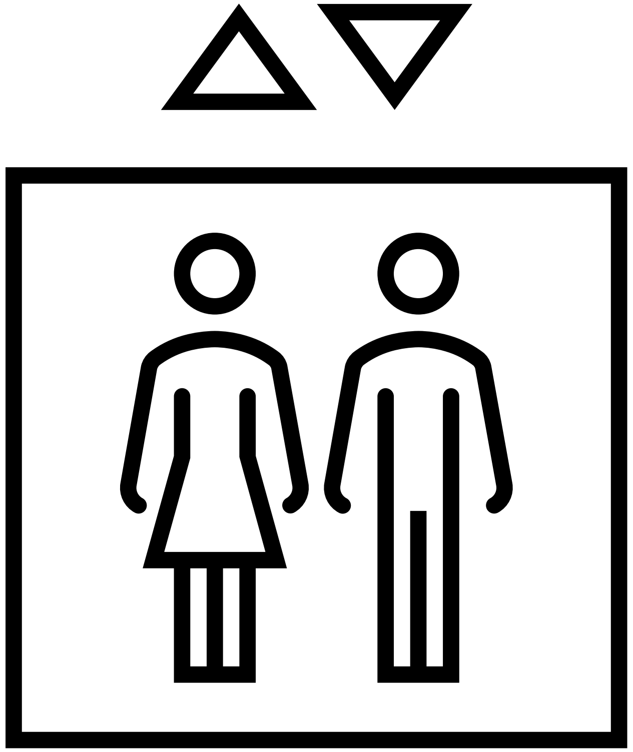
Let's take an example

Elevators

- A convenient way to travel up or down in tall buildings
- Nowadays, elevators are controlled by computer programs
- Ideal example since:
  - Everyone is familiar with them
  - We think we know how they work
  - They are relatively simple

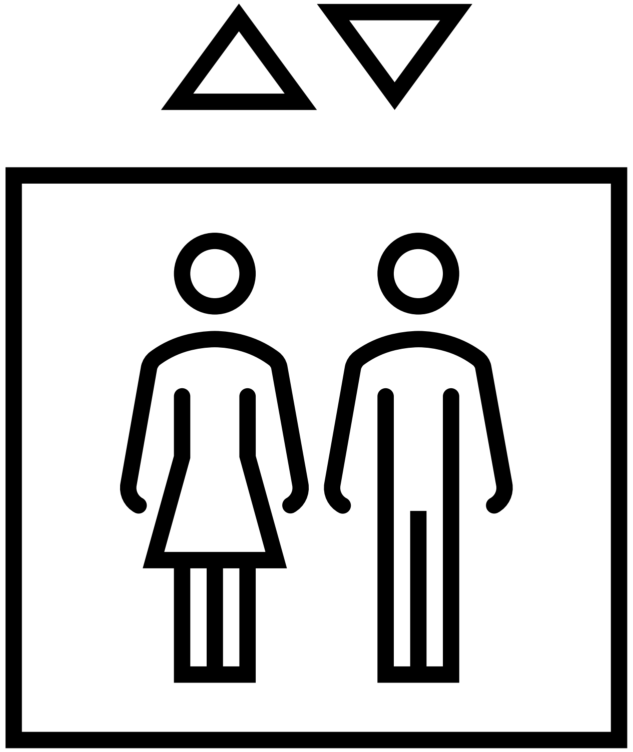
How does an elevator trip look like?

Type on the chat!



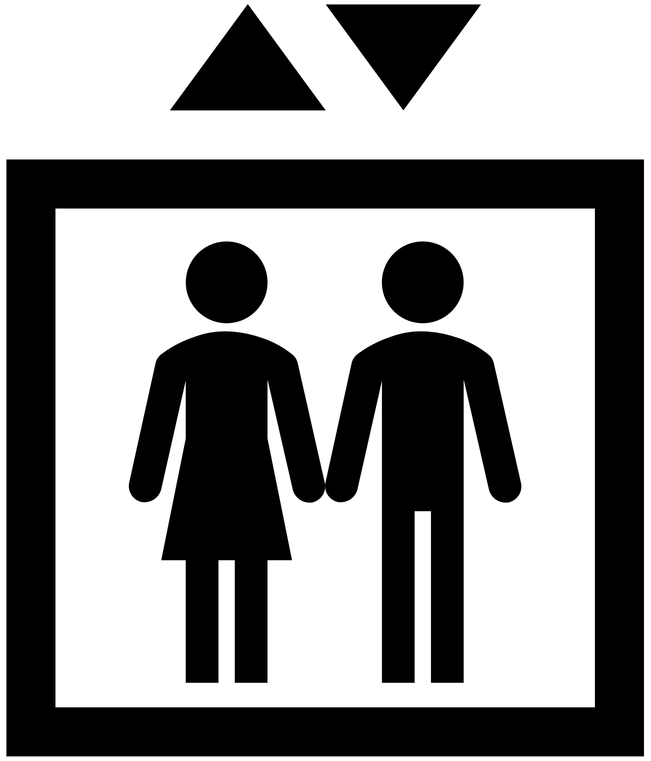
How does an elevator trip look like?

1. Call an elevator
2. It arrives
3. Door opens
4. You enter
5. Select destination(s)
6. Door closes
7. Travel to the destination
8. Door opens
9. You exit



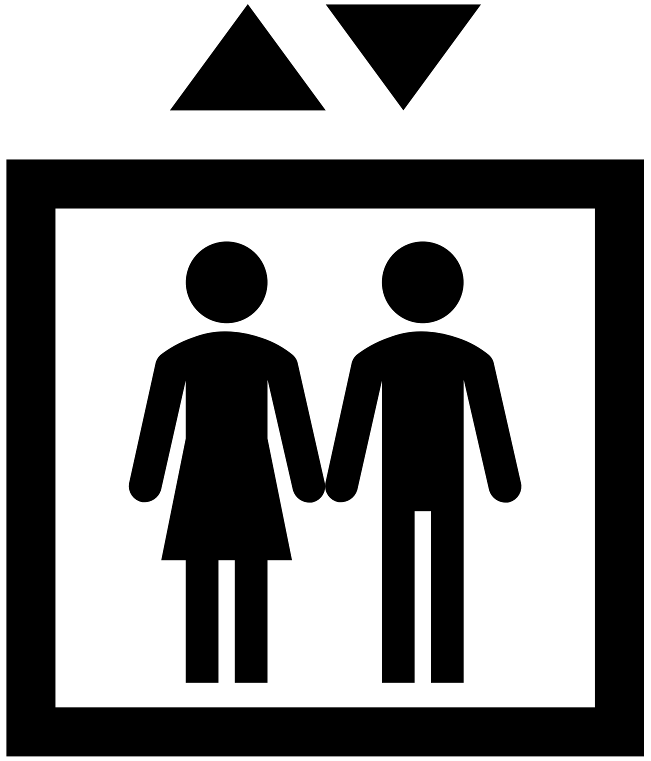
How can we model elevators?

Type on the chat!



How can we model elevators?

- Capacity
- Movement
- Attitude (direction)
- Door open/closed
- Load factor
- Buttons (in the car/on each floor)
- Speed
- Total number of elevators
- Number of floors covered



How can we model elevators?

Properties/attributes

- Capacity in kg (total/occupied)
- Movement direction (current and for next step)
- Speed
- Buttons pressed (in the car only)
- Position (which floor?)
- Lowest/Highest floor stop

Behaviour/functions

- Load/unload
- Stop, move up, move down, open door, close door
- Accelerate/break (rules depend on distance as well)
- Press button
- Set program to visit certain floors

A few solutions using different models

Simplistic

- Position (floor index)
- Movement direction
- Floors covered
- Program

Brute force

- Position (floor index)
- Movement direction
- Floors covered
- Program
- Speed

Democratic

- Position (floor index)
- Movement direction
- Floors covered
- Program
- Speed
- Buttons pressed
- Elevators called

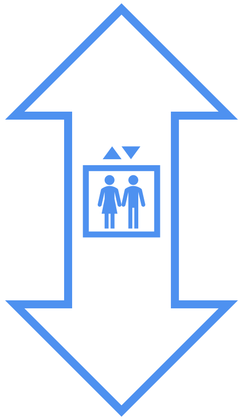
Smart

- Position (floor index)
- Movement direction
- Floors covered
- Program
- Speed
- Buttons pressed
- Elevators called
- Load factor and people waiting
- Synch. program

Where is my abstraction?

I want a refund!

Abstract



Detailed

Level of abstraction

The amount of complexity by which a system is viewed or programmed. The higher the level, the less detail. The lower the level, the more detail.
The highest level of abstraction is the entire system. The next level would be a handful of components, and so on, while the lowest level could be millions of objects.

PC Mag Encyclopedia

Abstraction for People

Simple

- Next destination

Self-conscious

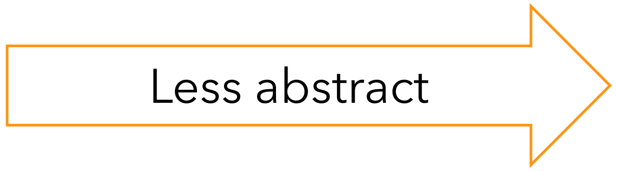
- Next destination
- Position (floor index)
- Waiting since

Unique

- Next destination
- Position (floor index)
- Waiting since
- Physical size/weight
- Preferences
- Behaviour

Special

- Next destination
- Position (floor index)
- Waiting since
- Physical size/weight
- Preferences
- Behaviour
- Can carry items (parcel/equipment)



Abstraction for elevators

Simplistic

- Position (floor index)
- Movement direction
- Floors covered
- Program

Brute force

- Position (floor index)
- Movement direction
- Floors covered
- Program
- Speed

Democratic

- Position (floor index)
- Movement direction
- Floors covered
- Program
- Speed
- Buttons pressed
- Elevators called

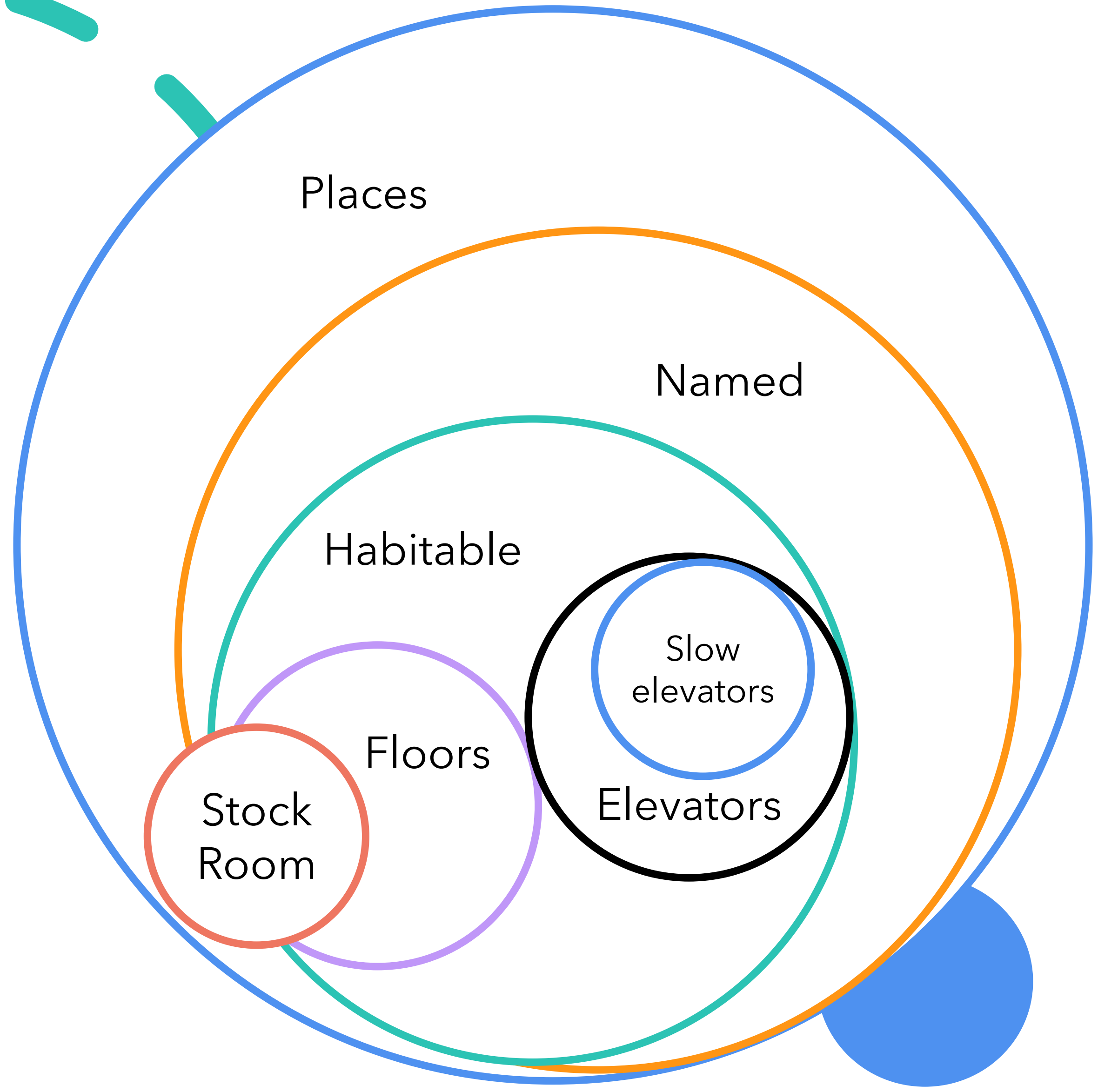
Smart

- Position (floor index)
- Movement direction
- Floors covered
- Program
- Speed
- Buttons pressed
- Elevators called
- Load factor and people waiting
- Synch. program

Less abstract

Can we go deeper?

- There is a lot in common between elevators and floors (name, floor index, population)
- But elevators can move
- Some can move with different speeds
- Stock room is special (it is where people spawn)



Enough of the slides!

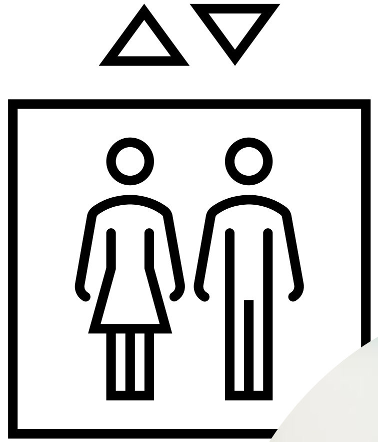
How will we get to the code?

We need requirements (or something similar)

"I want all my elevators to transport my employees as fast as possible"

What is the problem with this?

Type on the chat!



The requirements (questions/remarks)

- Not precise
  - What does “all” mean?
  - What makes the elevator faster when it comes to transporting employees?
    - Actual speed?
    - Shorter maximum waiting time?
    - Shorter average waiting time?
    - Should it be faster for each employee individually or more efficient in transporting the whole group?
- How can we identify and measure success?

The requirements (an example)

User story

As the elevator operator, I want the elevators installed in the Forest Offices to *minimize the average time* spent on *waiting* for the elevator *and travelling* in the elevator while people are using them, as *measured on a typical day*.

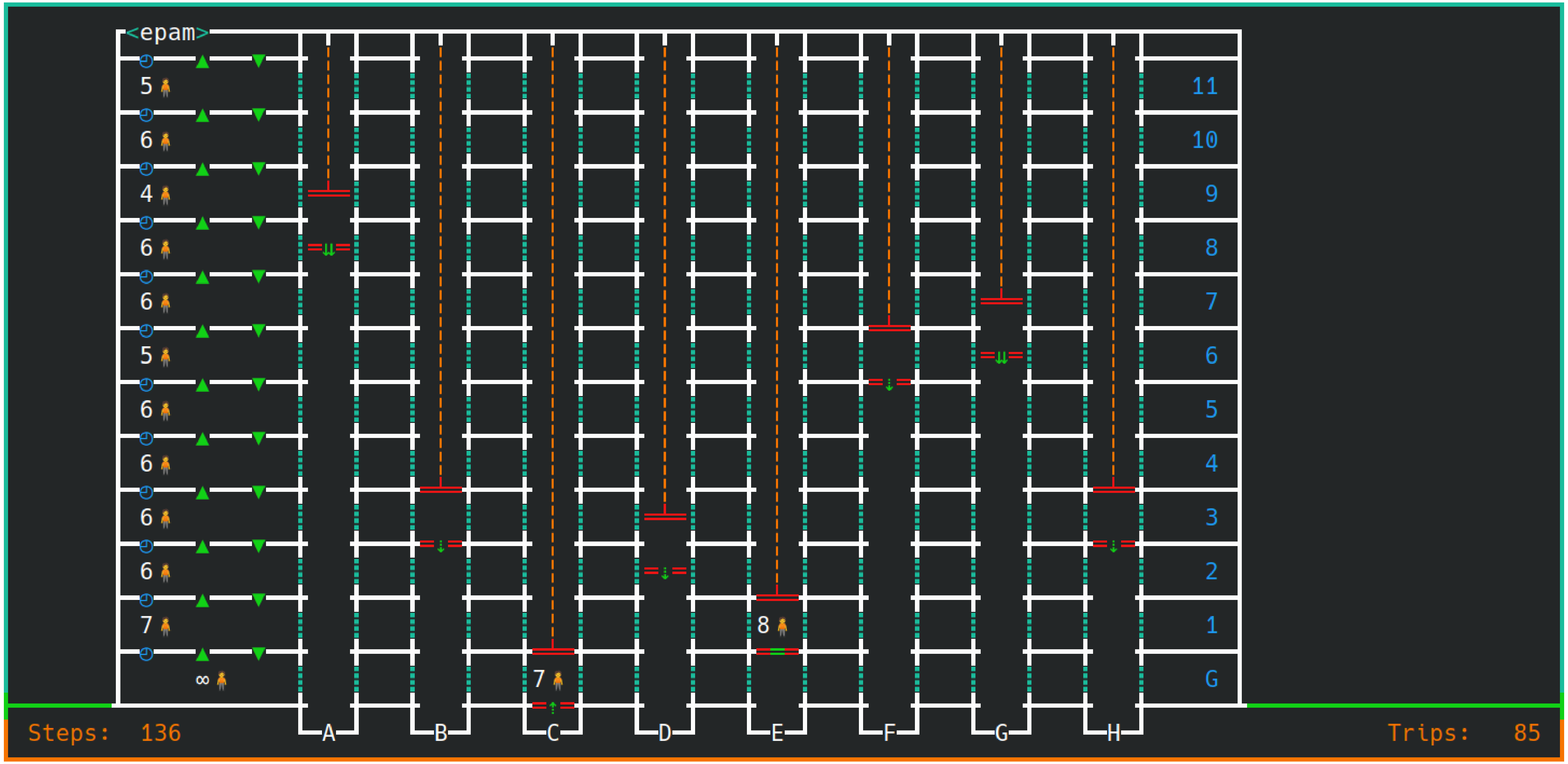
Acceptance criteria

Elevator controller implementation is changed to:

1. only stop elevators on a floor if:
    1. the button of that floor in the elevator car is pressed; or
    2. the caller button on the floor is pressed, and the elevator is:
        1. idle (no buttons pressed in the car); or
        2. moving in the direction of the floor where the button is pressed, and can stop safely on the floor
2. select the idle elevator that can arrive to a floor under the least amount of time when more than one elevators are idle
3. avoid sending more elevators to a floor than necessary in order to transport waiting persons

The simulation

at last!



```
Steps:  136                                                    Trips:  85
```

The “modern UI” I decided to use, appears to be older than I am, but it is from 2021.

About the project

Project home

- https://github.com/nagyesta/next-level-abstraction

Limitations

- Heavily uses UTF-8 characters (Windows terminal will show '?'-s)
- Needs your terminal to understand ANSI control sequences to move to the top left corner

SCAN ME

Thank you!

Istvan Nagy

Website:

https://nagyesta.github.io