Elevator 51

Elevator Control System Demo

R. Cotrina, K. Day, J. Del Prete, M. Frystacky

Plan

Goal

- Create a elevator algorithm that is:
    - Time-efficient
    - Energy-efficient
    - Scalable

Tools Used

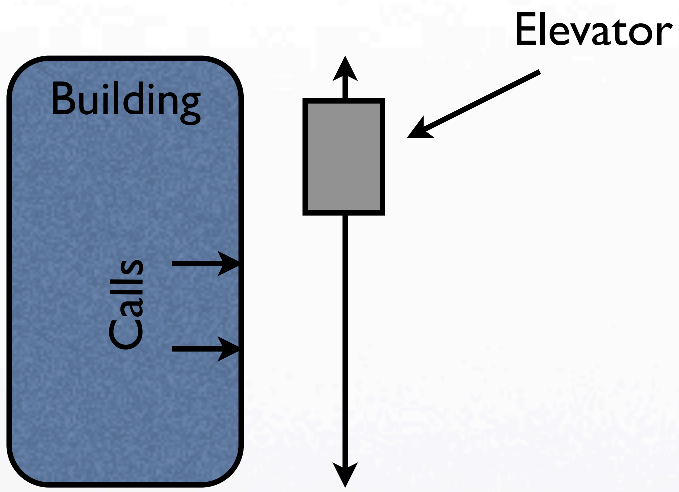
- Eclipse IDE
  - Development
- Git Hub
  - Sharing code, version control
- Asana App
  - Communicating, scheduling, setting goals

Plan

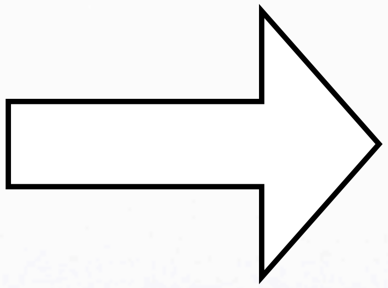
- Create "basic" algorithm
- Refine to make "intelligent" algorithm
- Compare "basic" and "intelligent" solutions by:
  - Average wait time for person
  - Average distance traveled by elevator

Basic vs. Intelligent

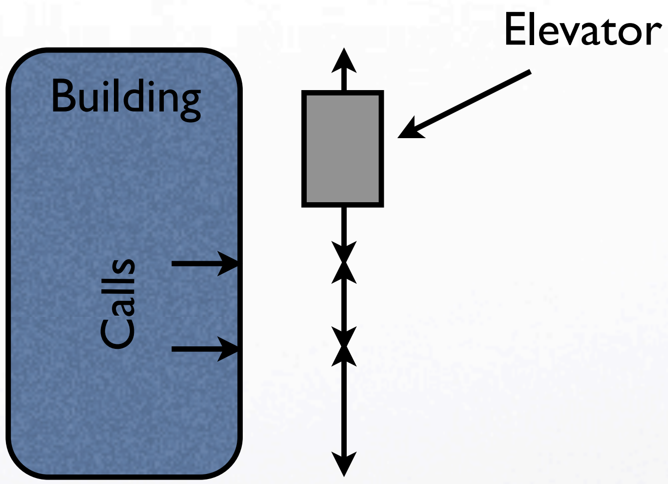
"basic"



Elevator travels in a basic pattern, just goes up and down
- Slow
- Inefficient



"intelligent"



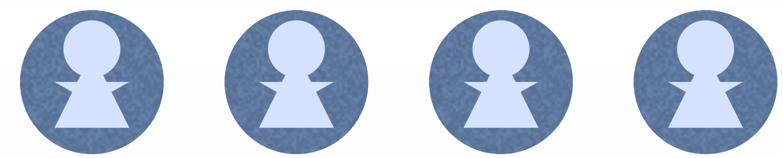
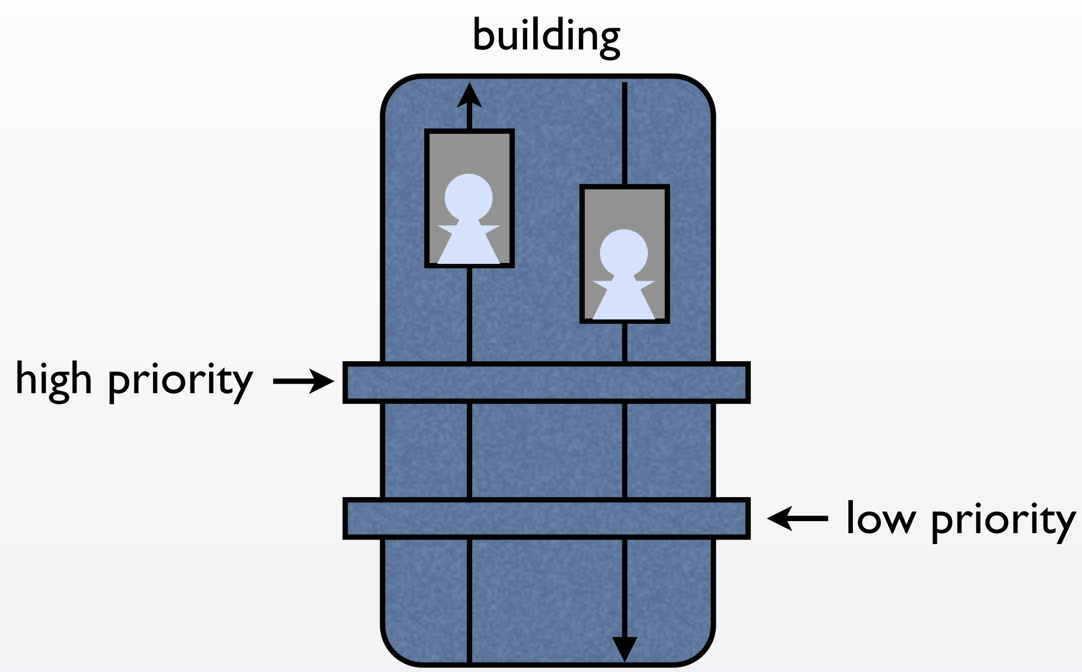
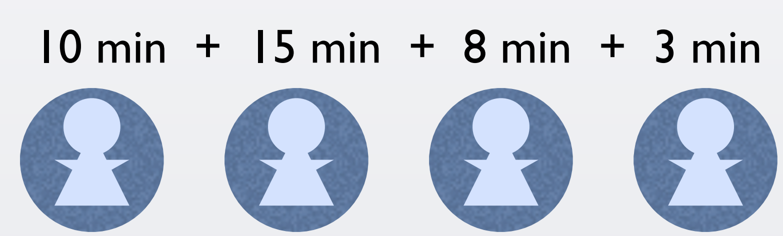
Elevator changes direction intelligently
- Faster
- Travels less distance
- Description on following slides

Intelligent Techniques

- **Floor fields** - Calculate priority of floors to travel to first
- **Elevator priorities** - Find which elevators are best for certain floors
- **Elevator fields** - Pick up passengers normally skipped

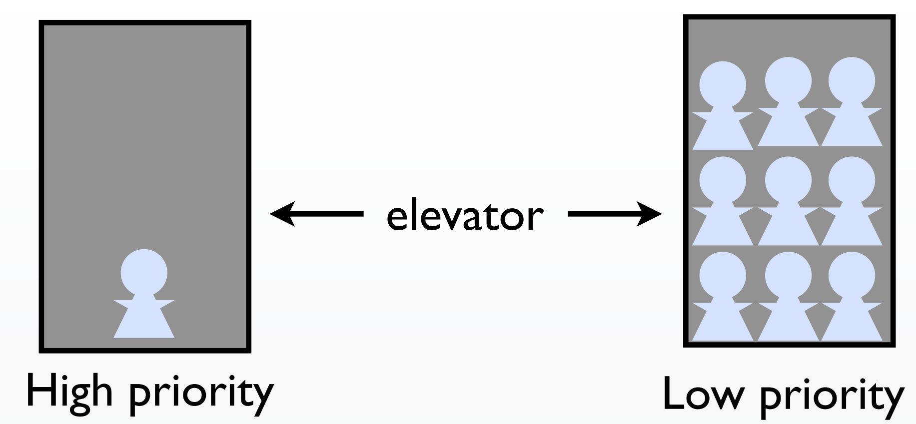
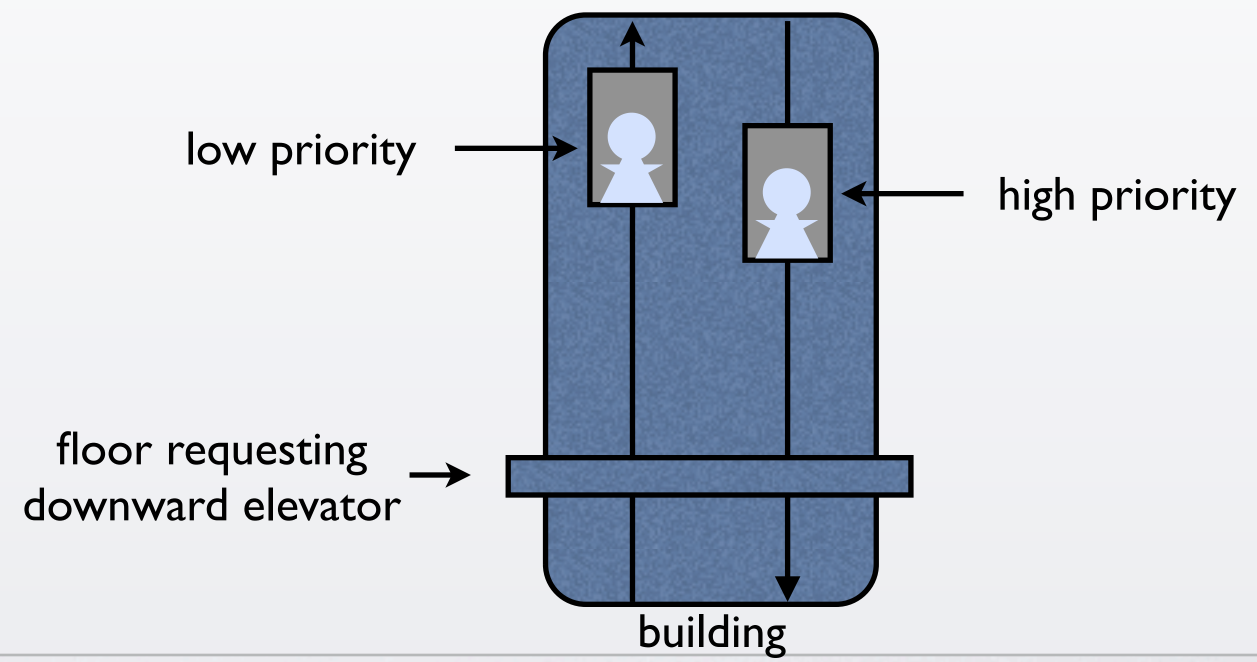
Floor Priority Factors

Which floor/call should be serviced first?

Number of people at that floor	
Proximity to elevators	
Wait time of people at the floor	

Elevator Priority Factors

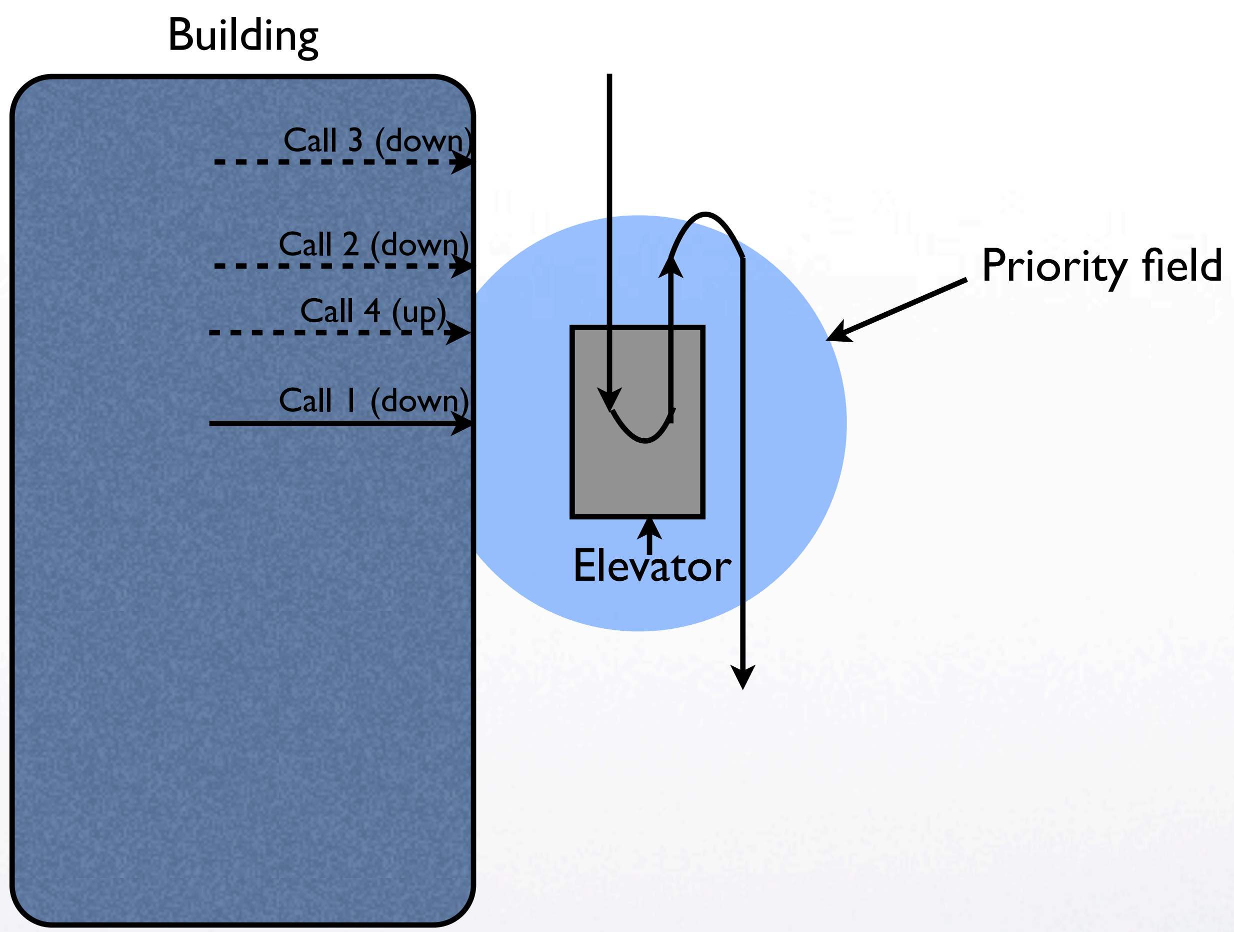
Which elevator should be used to handle this call?

How full the elevator is	
Direction elevator is traveling	

Elevator Fields

- Elevators are surrounded by a “field”
  - Calls within the field going in the same direction take top priority
  - Prevents wasting people’s time if they miss an elevator by a few floors
  - Field will “dissipate” after use for a certain time interval to prevent abuse

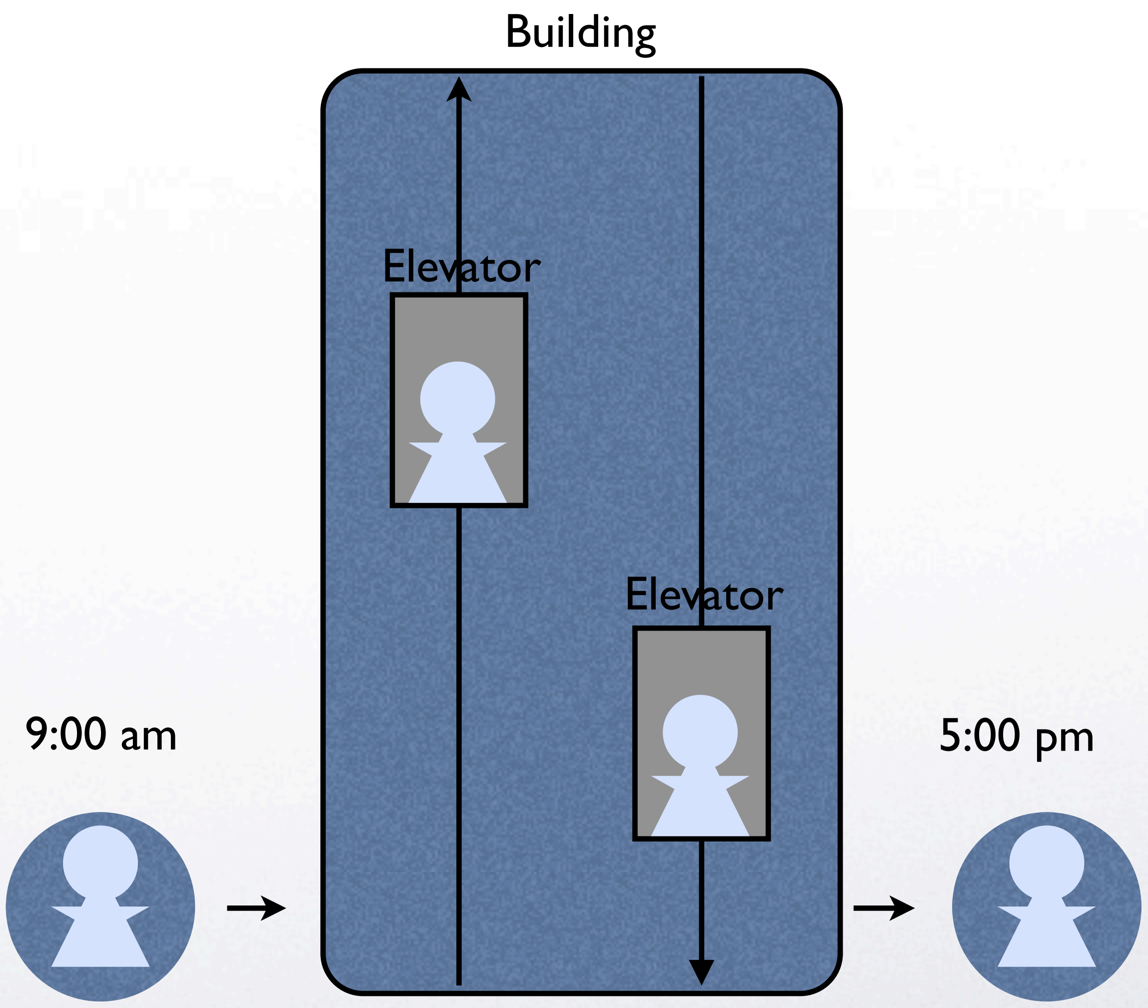
Example



The calls occur in the order <1, 2, 3, 4>.
The elevator will go back for call 2, but not for call 3 or 4.

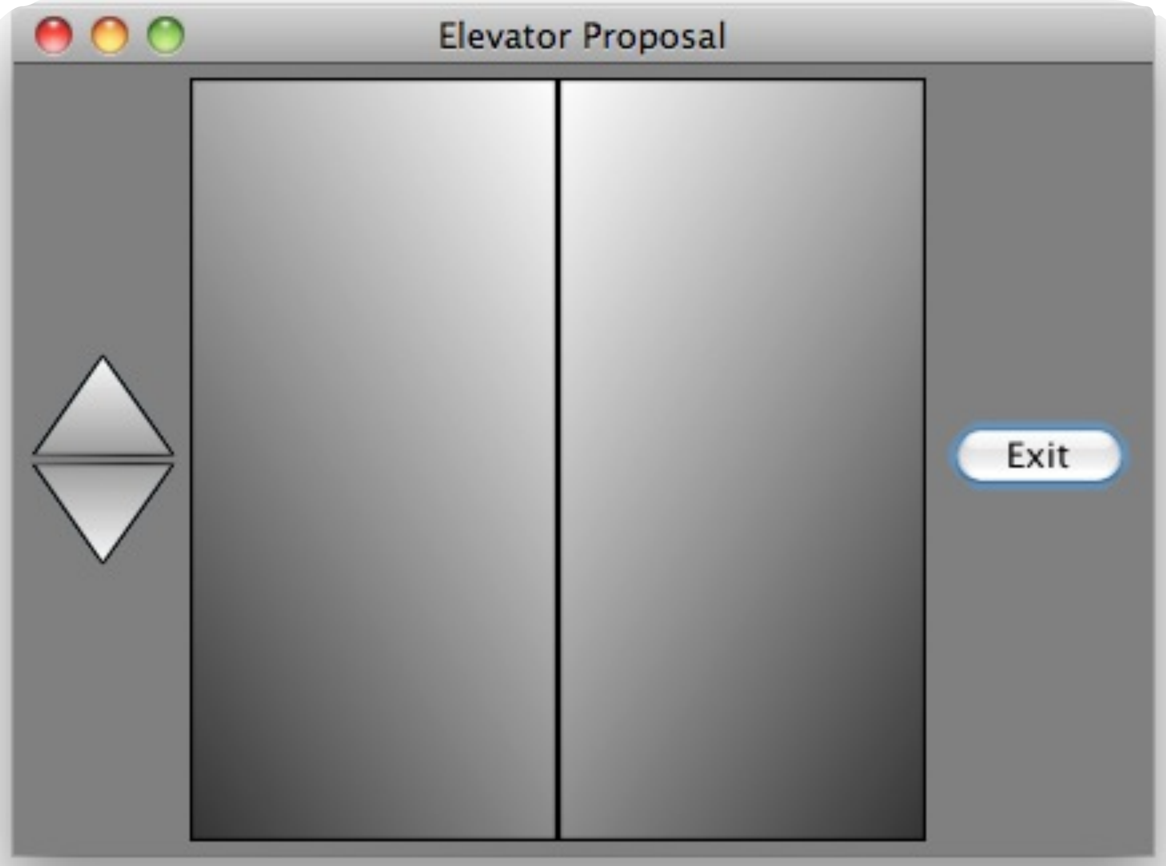
Testing

- Simulate a typical work day
  - People arrive in the morning
  - People move at lunchtime
  - People leave in the evening
  - Cleaning staff come in at night
  - People move around throughout the day
- Simulation lasts for 24 hours



GUI Plan

- Initially planned for two GUIs
    - “User experience” - what a person using the elevator will see
    - “Building view” - positions of all elevators in the building
- “User experience” cancelled due to time issues



Screenshot of scrapped “user experience” GUI.
Buttons lit up and the doors moved.

Solution

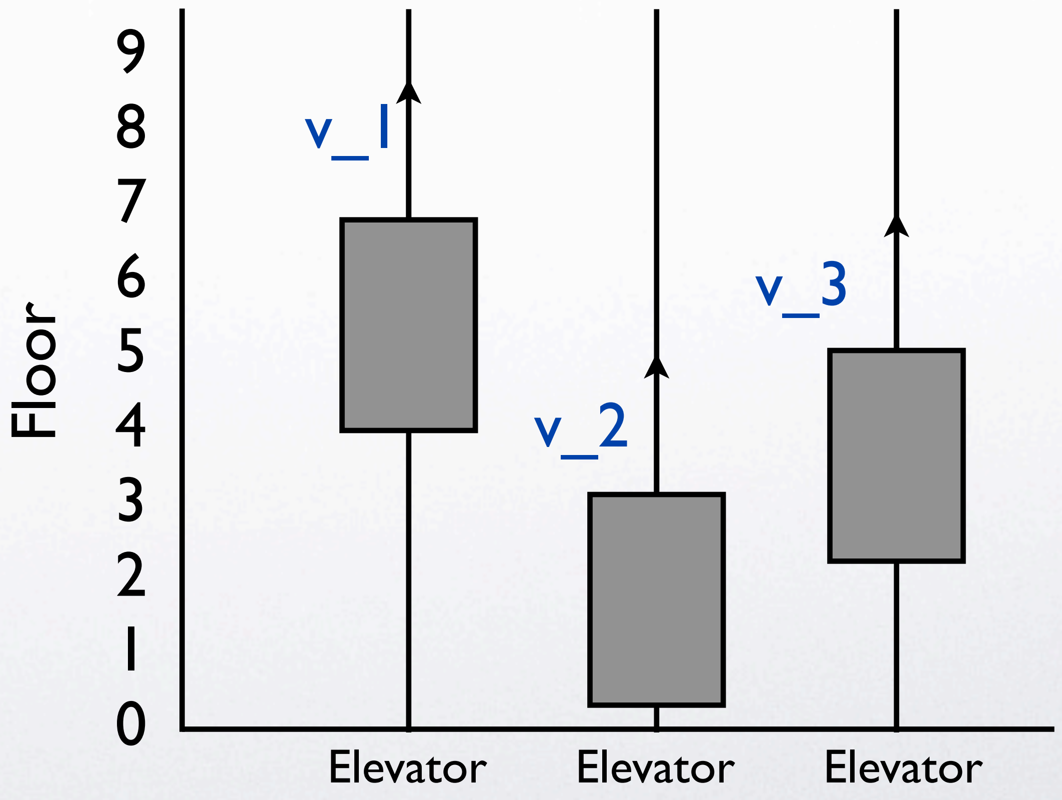
Who Did What

Name	Algorithm	Test Cases	GUI	Presentation & Data Analysis
Roger	X			
Kim			X	X
Joe	X	X		
Michal	X			

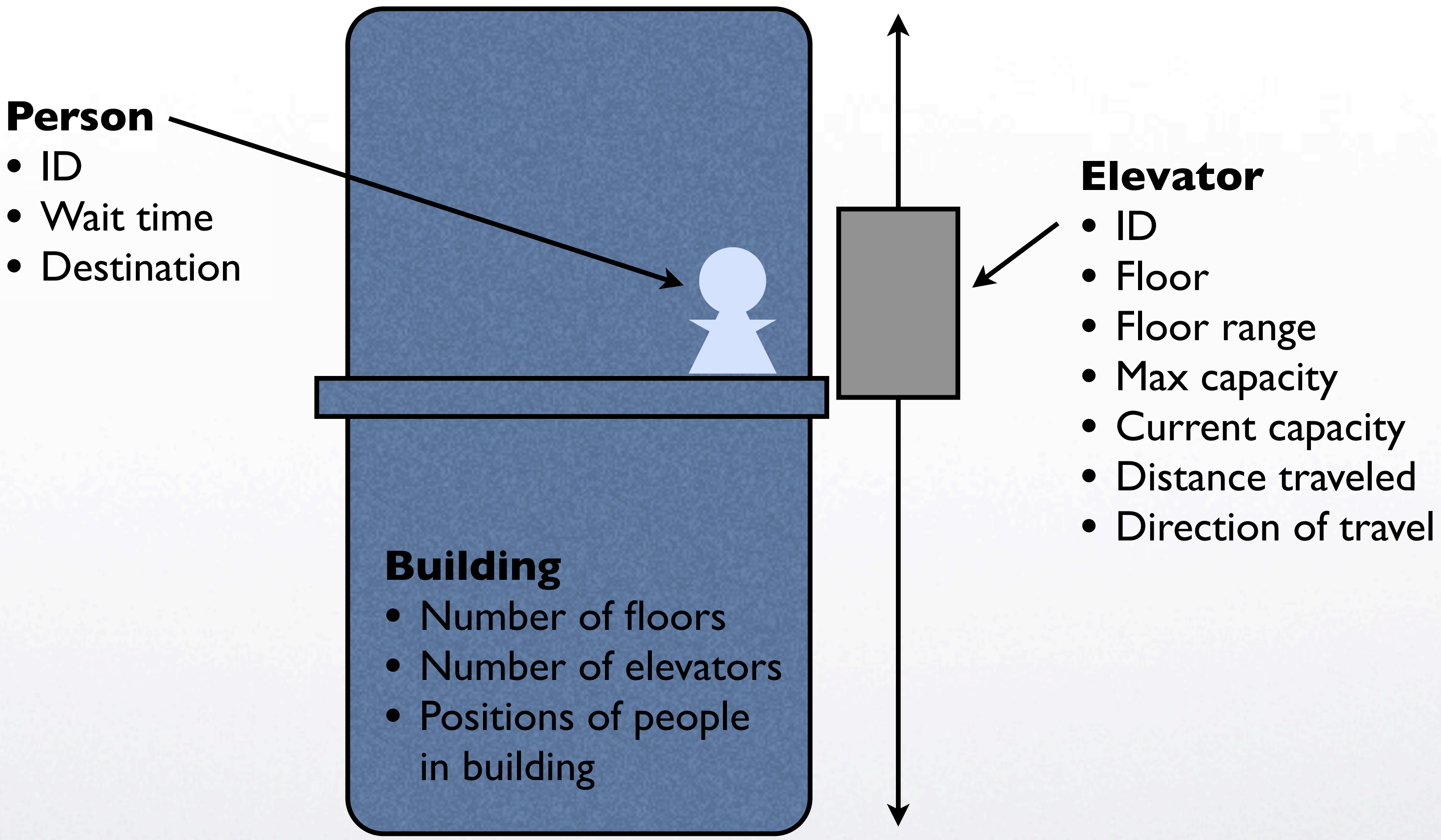
Assumptions

- All elevators move at the same speed
- All floor indexes start at 0
- Picking up/dropping off passengers is instantaneous

v_1 = v_2 = v_3 = 1 floor/minute

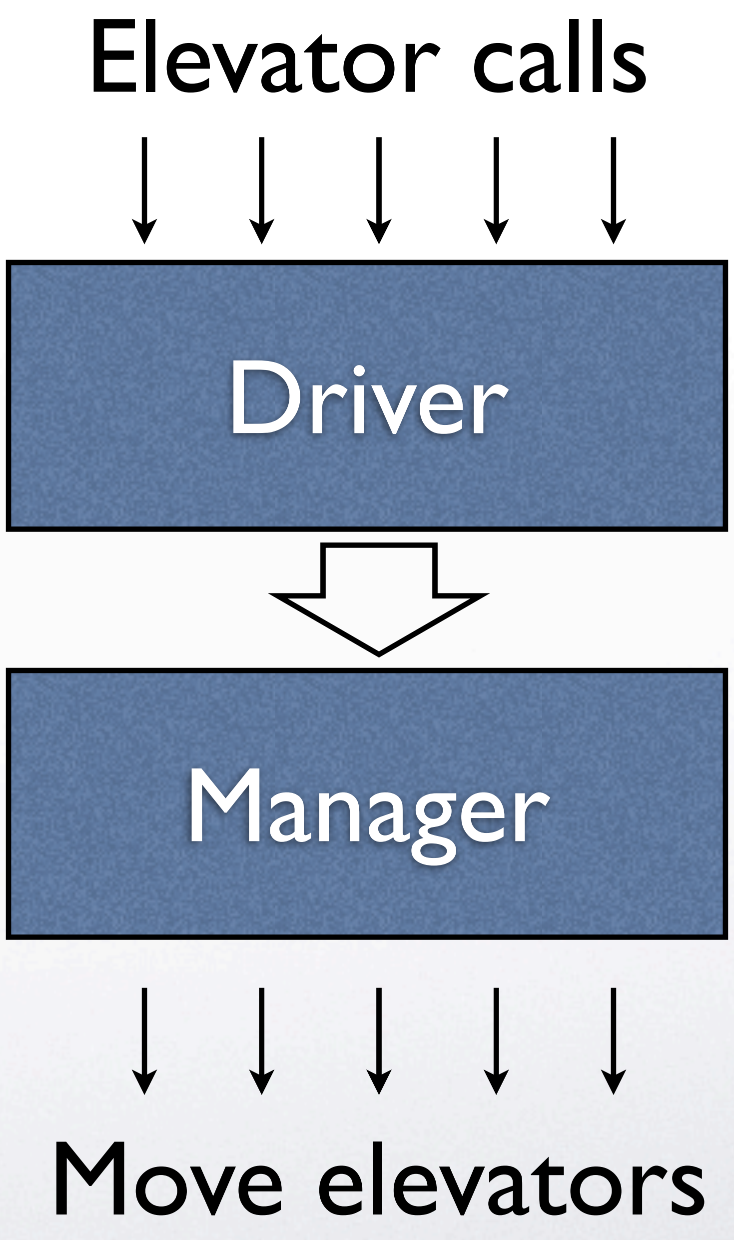


Classes: Objects



Classes: Main

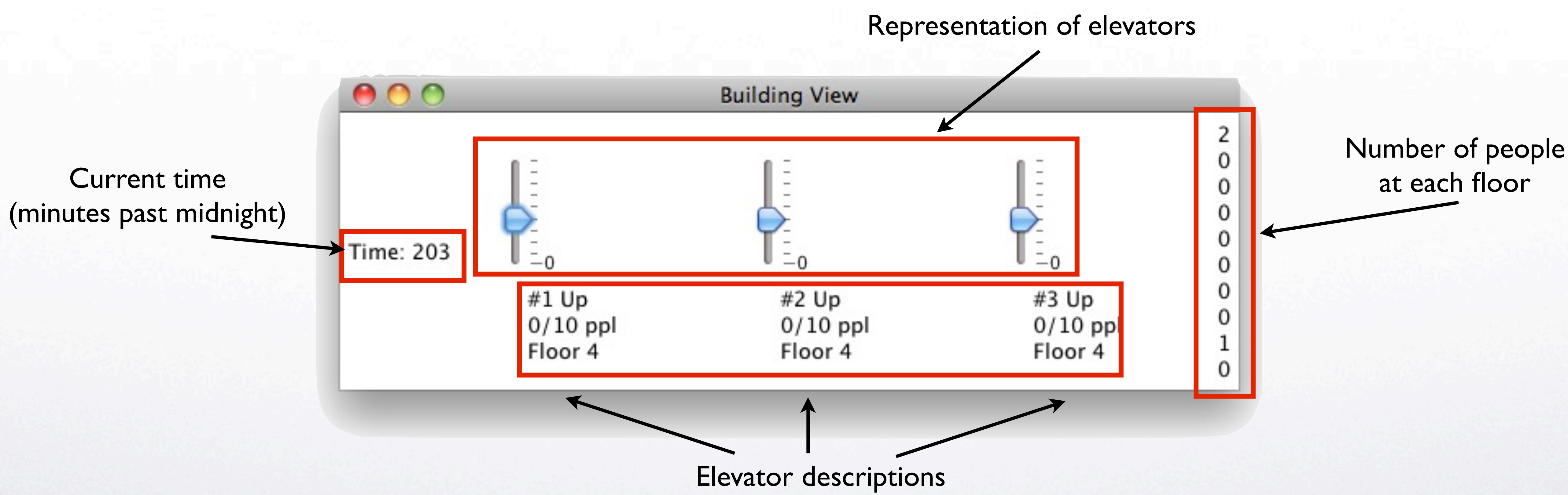
- Driver - Interprets the elevator calls and sends the requests to the manager
- Elevator Manager - Evaluates the situation and moves the elevators according to our algorithm



Classes: Other

- CustomQueue - Custom comparators for comparing elevators and floors
- BuildingSwing - Creates a GUI to represent the whole building
- ElevatorSlider - Creates a Panel to represent a single elevator in the GUI

GUI Screenshot



Fixed Issues

- People trapped inside elevators at the end of the day
- Elevators spontaneously teleporting between floors
- Elevators shooting through the roof, reaching floor 9000+ in a building with 10 floors
- Elevators having a negative number of passengers (ghosts?)

Our Implementation

Conclusion

Findings

	Avg Wait Time	Avg Dist Traveled
Basic		
Intelligent		

Discussion

- Intelligent algorithm improved upon basic algorithm by