The Software Project Day

Curtis Burtener, Kristen Mills, Emma Nelson, Ian Salitrynski

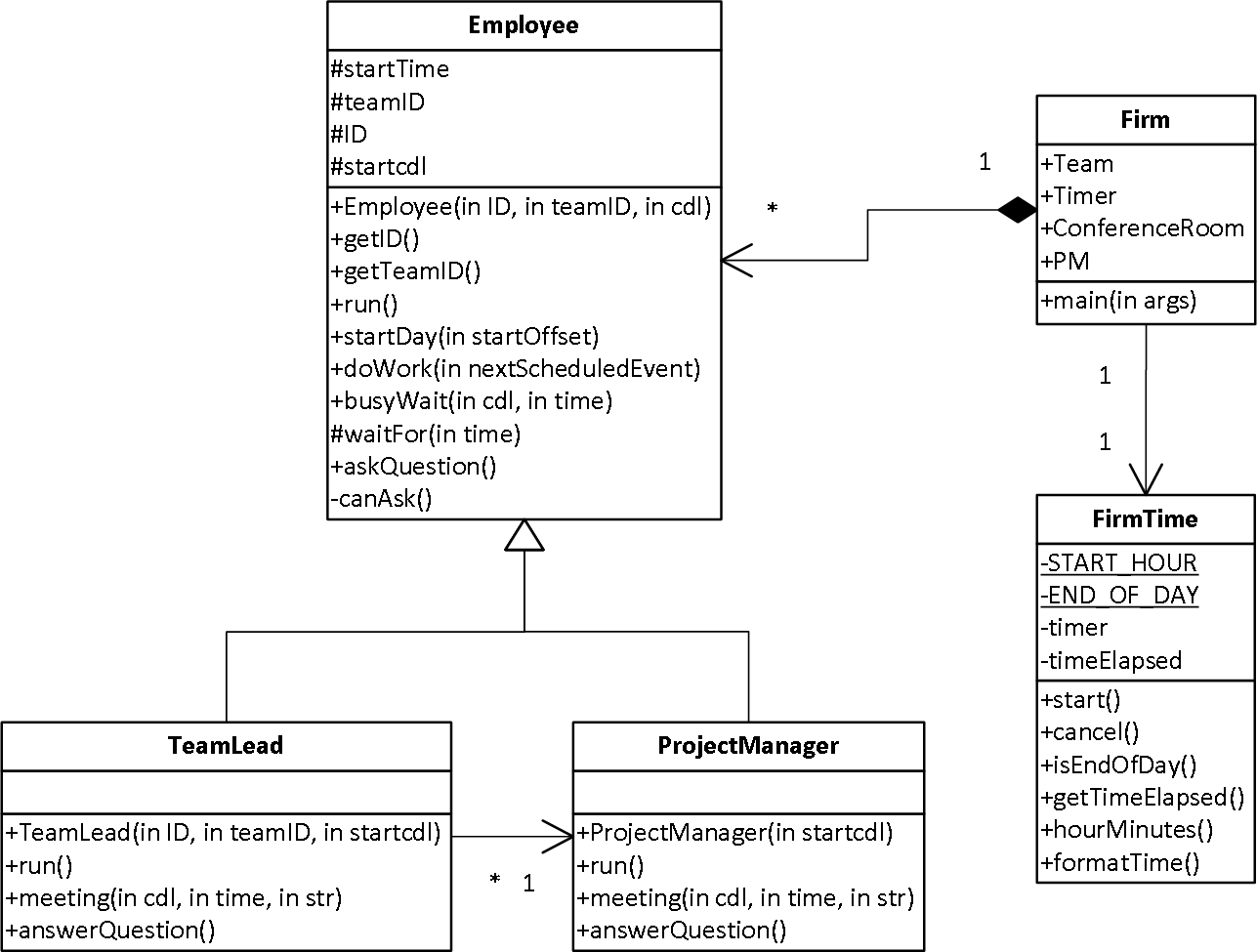
# Assumptions

We made a few different assumptions for this project. We decided no new questions can be asked after 4:00 PM, question asking is done by random number generation every minute with a .25% chance of success, and all employees start lunch between 11:00 AM and 12:30PM.

# Final Design

Our finished design is very simple. It relies heavily on concurrent features in Java. The relationships between different types of employees makes use of inheritance, and all employees, except for the Project Manager, as stored in a two dimensional array which represents the firm structure. The time is handled by a modified version of Java’s Timer class. The main method is in the Firm Class, which is the driver of the program and holds references to everything needed to execute the simulation.

## Class Diagram



## Data Structures Used

A two dimensional array was used to represent the company structure and collect all employees for easy access. Each internal array represents a team with the team lead at the zero index. This structure makes it really simple for any team member to find their team lead without holding a direct reference to him or her because they can just index the firm array by their team number and grab the zero index of the returned array.

# Alternative Designs Considered

Originally our design resembled a composite pattern with the firm being made up of teams and employees (team members and team leads), where the employees had to be a part of a sub-team. We did not get far enough with the design to work out how to handle time beyond the use of a separate class. The Project Manager was also an object outside of the composite structure. We threw out this plan the meeting after we created it, because it complicated the project more than necessary and we couldn’t visualize it coming together smoothly. The chosen design also got rid of the extra overhead involved in implementing a composite pattern and making sure it worked for our system.

## Class Diagram

# Results of Experiments

Overall very few difficulties occurred in the actual development of the system. We hit a few road blocks in design, but we worked that out at the following meeting as noted above. On the technical side, many of our roadblocks involved timing, deadlock prevention, and synchronizing the threads during the day. The timing was fixed by using a starting countdown latch on all threads and by changing the interval that threads wait when doing work.

The conditions for deadlock to occur existed in our code in a synchronized method near the end of the day. All employees countdown and await the 4:00 meeting countdown latch. Deadlock could occur if the Project Manager entered this method, holding its own lock, while an employee tried to ask a question. The employee would wait for the Project Manager to release its lock, but that would never occur until the employee's question was answered. We solved this by removing the synchronization.

Synchronizing the threads during the day became complicated as more countdown latches were needed at different parts of the day. We considered using barriers; however, many of the activities in the day happen between distinct groups of employees once a day. This increased coupling in our code, as employees need to know all the latches they will use in a day.

We did not have roadblocks with providing access to things like the conference room, Team Leads, and the Project Manager. This is because we carefully designed the project so as to easily allow synchronized access to parts requiring it.