The Software Project Day

Curtis Burtener, Kristen Mills, Emma Nelson, Ian Salitrynski

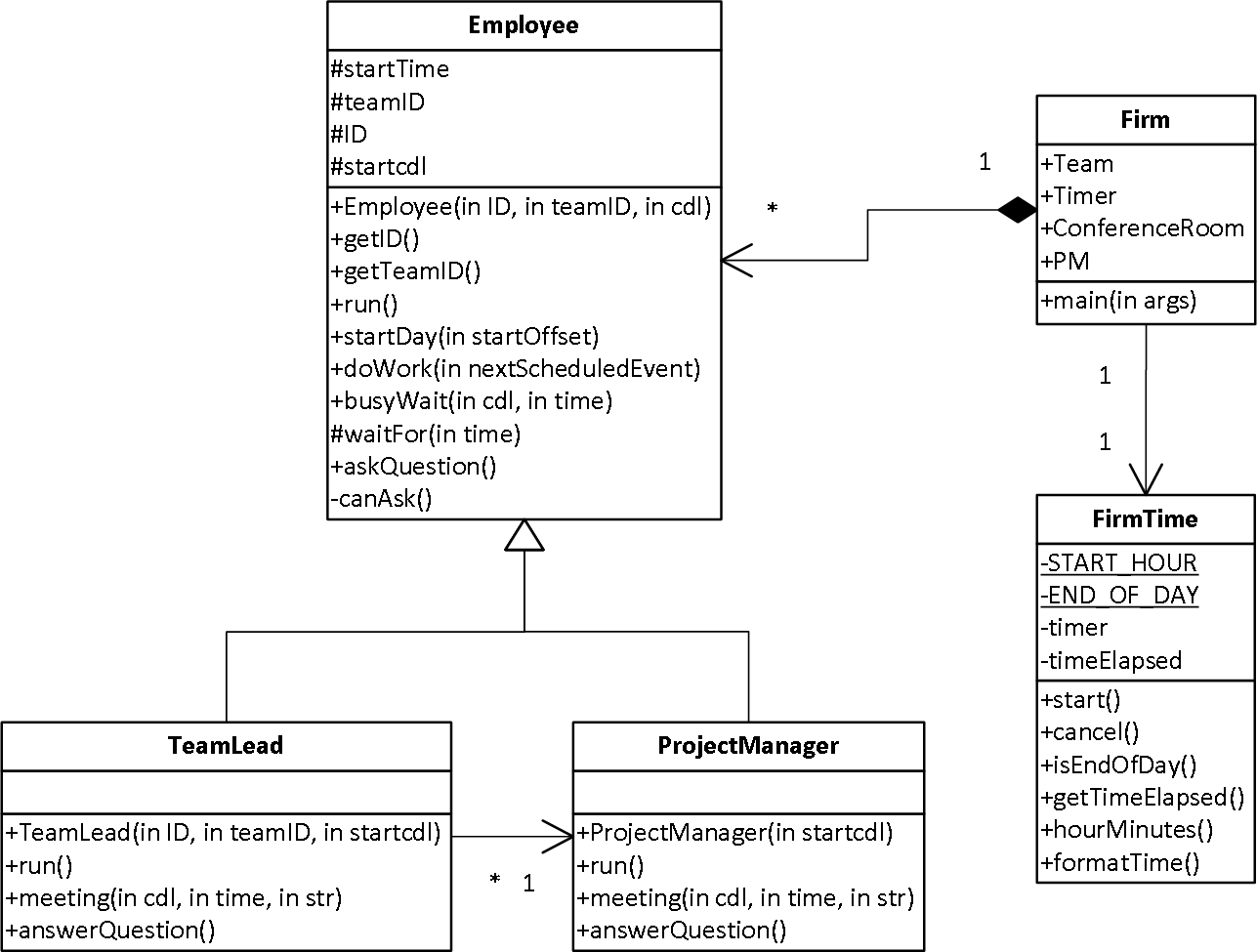
# Assumptions

We made a couple of different assumptions for this project. We decided that the team leads will wait for all members of their team to leave before leaving themselves, 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

Curtis will probably write this…