Project 6 Status Summary

Team Members: Angle Dong, Sam Feig, Elly Landrum
Project Title: Flashpoint Lite
Github Link: https://github.com/AngelD2000/OOAD/tree/master/Proj6RestartReal

Work Done:

- View (Angel)
  - Flashpoint: The top most level of frontend like setting the scene and stage for objects to be displayed on.
  - Display: Takes in a square and checks whether there will be anything displayed on the square.
  - ViewManager: The controller for frontend, this is where stage, scene are declared, and any frontend functions that are needed. Like draw map, edge, display the current stats etc.
- Controller (Elly)
  - Game: Game controls the general logic for the game. This includes making heavy use of Building, FireLogic, and FirefighterLogic. Game is also responsible for handling orchestration of end of turn behaviour.
  - Building: Building contains the information on POI (points of interest) and victims. It handles the logic of flipping POI into victims or blanks, rescuing victims, killing victims, and placing new POI. Building also keeps track of end of game information for victims saved and rescued.
  - FireLogic: Calculates the effects of advancing fire. Advancing fire also causes explosions when applicable. MapBuilder can also use explosion logic to set up the game.
  - FirefighterLogic: Controls the logic of a firefighters turn. Can calculate legal action for a firefighter, and the result of a firefighter action. FirefighterLogic also tracks how many actions a firefighter has left
  - Company: Company is a collection of firefighters. The current implementation of company tracks which firefighter is active (is taking their turn). It also handles accessing the content/methods of each firefighter.
  - Firefighters: Each firefighter has an image in addition to functions to get and move the location of the firefighter.
- Model (Sam)
  - Square: Class filled out and functions stubbed for decorators, logic for edges is implemented to get an edge in a specific direction.
  - BaseSquare, FireSquare, SmokeSquare, OutsideSquare, POISquare: Created decorators for types of squares and added their specific functionality through overloading of stubbed Square functions. Equality checking for squares.
  - Map: Created Map as an iterator of Square objects from the 2d array map. Functions for getting the position of a square or square at a position filled in. Added functionality for adjacency checking, neighbor retrieval, adjacent fire

checking, and edge checking. As part of being an iterator, it implements hasNext() and next() to go through all the squares in the map. Must call resetIterator() when done currently.

    - MapFactory: Creates a standard hard coded map with outside squares and edges

Changes and Issues:

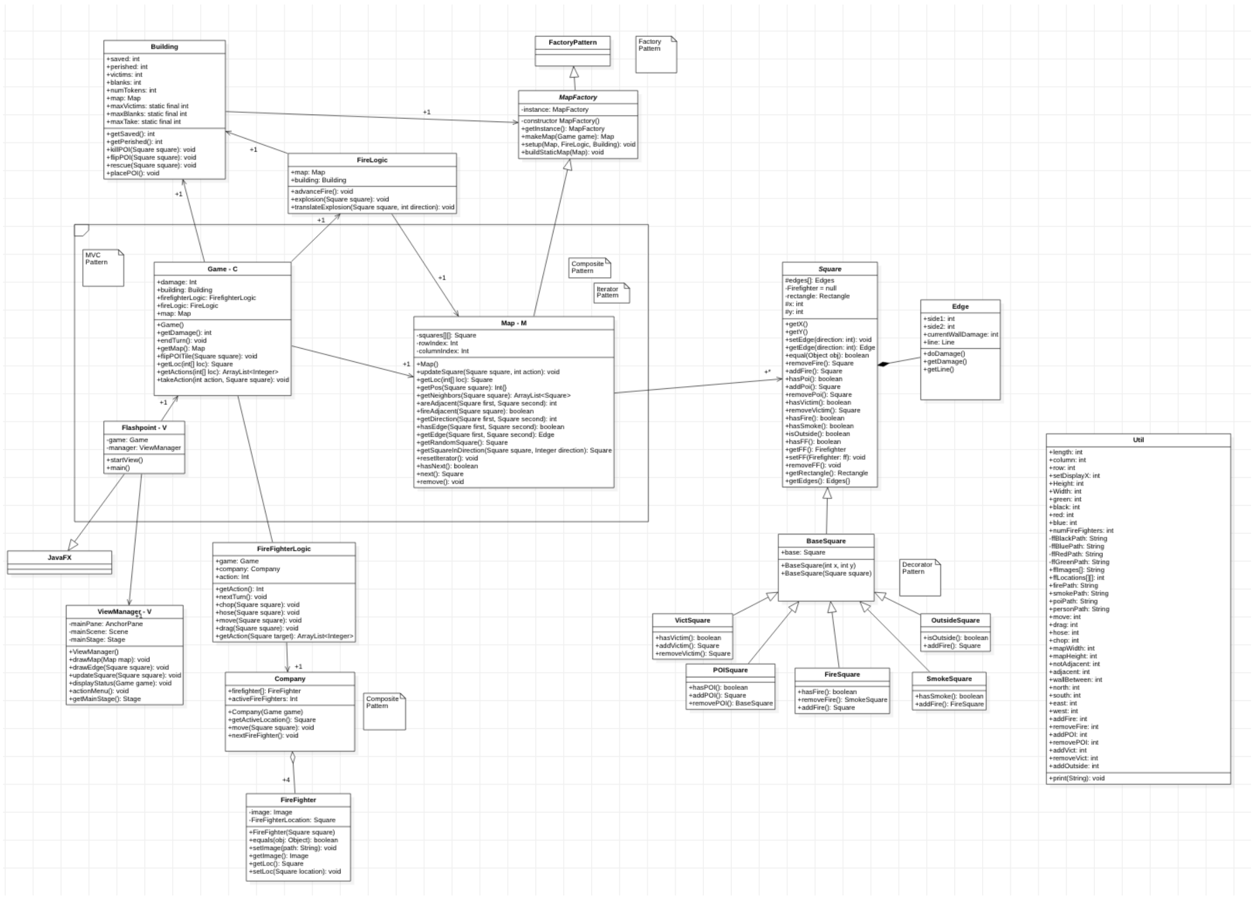
- Changes
    - Additional functions added to at least every class to handle some of the specific functionality needed and missed in initial design.
    - Functionality that was originally just placed in the Game object has been moved between FireLogic and FirefighterLogic for better cohesion/one class = one responsibility.
    - Firefighter Pool renamed to Company, and is instead a composite of firefighters.
    - Added a Display class to encapsulate the processing of each square so not all functionalities are in the controller class.
    - Made firefighter a type of decorator on the square, so firefighters, fire, poi, any object that is on a square will be a decorator for the frontend to replace the background of the square.
    - Both Square and Edge need a Javafx object to display. Rectangle and Line class are the two I was using.
- Issues/Challenges:
    - Connecting all of the components together
    - Using Maven, especially getting project to run on Windows and Mac
    - Displaying items
    - Squares need to have their decoration pass through to enable multiple decorators

Patterns:

- MVC: The MVC pattern has been super helpful for helping us keep our work organized. It makes it easier to conceive whether many actions are a responsibility of a class, or if it should be delegated elsewhere. Thinking of the parts in the MVC way also allowed us to better split our work.
- Composite: Composite is used both for the Map and Company Objects. It's made it a lot easier to think about each object as a collection of objects. Thinking about the object in this way made it easier to realize some of the functions necessary to pick an individual object from that collection.
- Decorator: BaseSquare implements the decorator and has decorators of types FireSquare, SmokeSquare, OutsideSquare, and POISquare that handle specific functionality for each type.
- Iterator: Implemented by the Map class to iterate over Square objects in the 2d map. Implements hasNext and next, the other two functions (remove and forEachRemaining) throw not implemented errors as they are not necessary for this application.
- Factory: MapFactory creates the entire map of the game each square at a time. This is really useful as the map is static and so has a lot of hardcoded components to it.

MapFactory is also an Eager Singleton as there is no reason for more than one factory to exist in the system.

Implemented Class Diagram:



Plan for Next Iteration:

In the next iteration, we would like to debug some of the things listed as issues above. We also need to add the logic and functionality for interacting with the game.