SWE 265P
Reverse Engineering and Modeling

Lecture 4

Reality

"…always find the particular 'point of interest' and then do chaining – chain backwards (who calls this – then who calls that – then who calls that) and if you iterate enough, you'll get back to main() at some point. You can also forward-chain all the way down into the utility libraries and the 'deepest' parts of the call stack, at least for the feature you are investigating." – Eric Dashofy [General Manager & Deputy CIO, The Aerospace Corporation]

Today

- Last week’s material
- Key expert practices
- Structural vs behavioral models
- UML in more detail
- In-class practice
- Consuelo Lopez (MuleSoft)

Last week's material

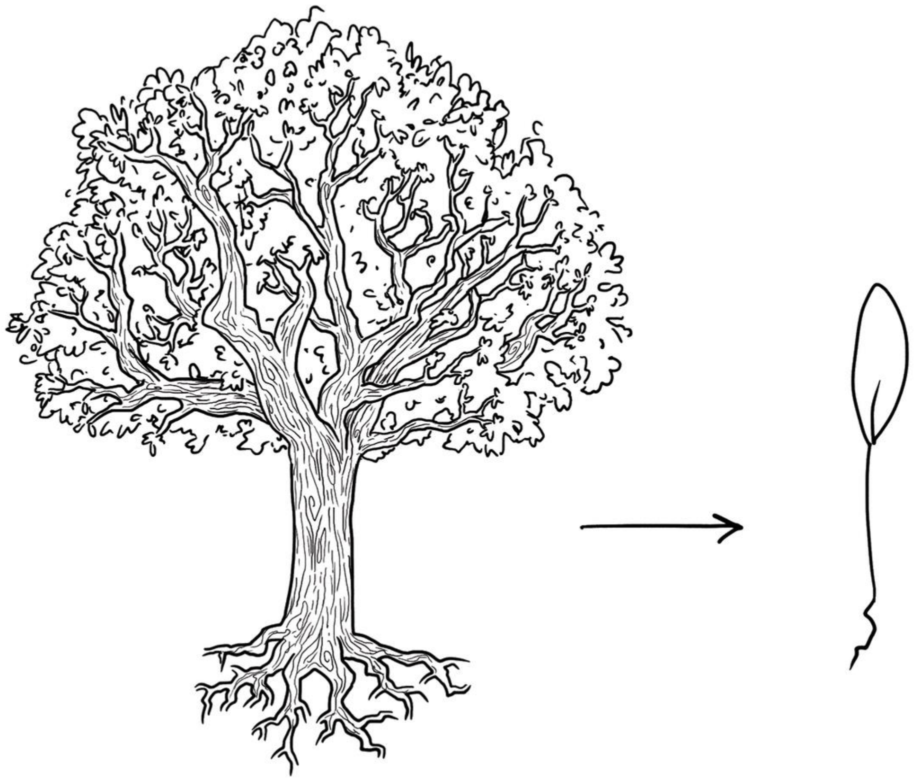
- Mental models
  - a representation of someone's thought process of how something works in the real world
  - individual, uncertain, selective, flexible, dependent
  - external versus internal (software)
  - limitations
- Externalizing mental models
  - where have we been & where do we still need to go templates
  - UML class diagrams
- Videos
- Any questions?

Last week's homework

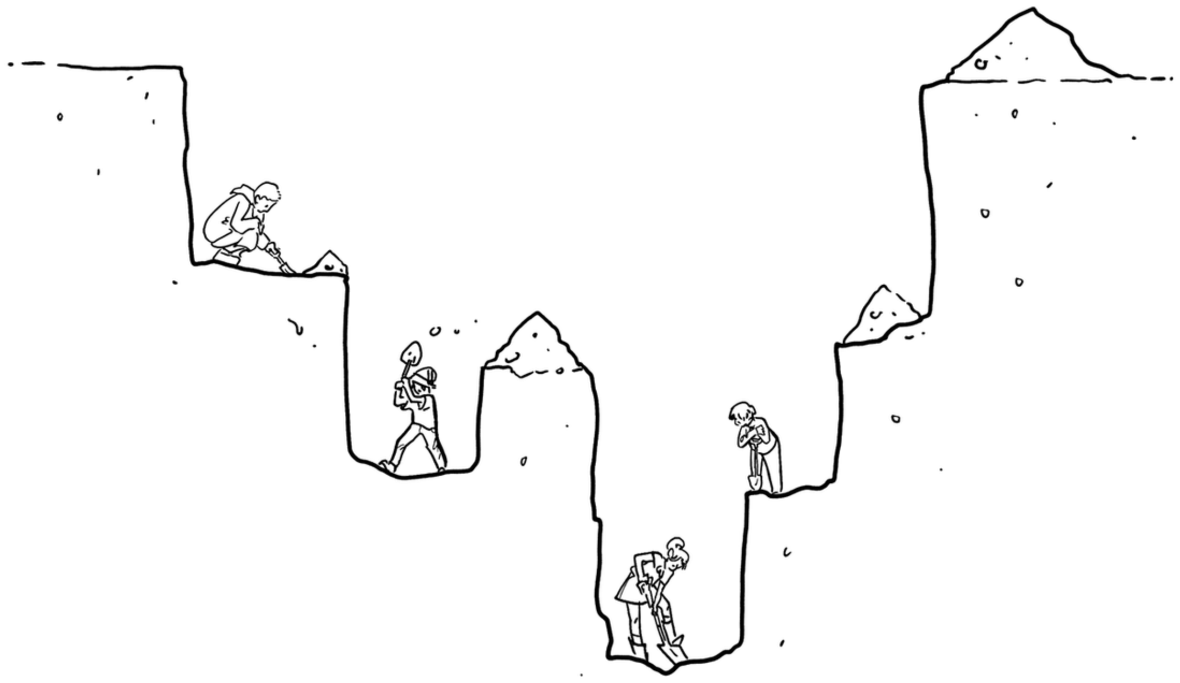
- Which features did you choose to locate in the code?
- How did you approach locating the features?
- How difficult was it? (Why?)
- How confident are you that you found everywhere in the code where the features are implemented?
- What is the value of the diagram that you now have at hand?
- Any questions?

Key expert practices

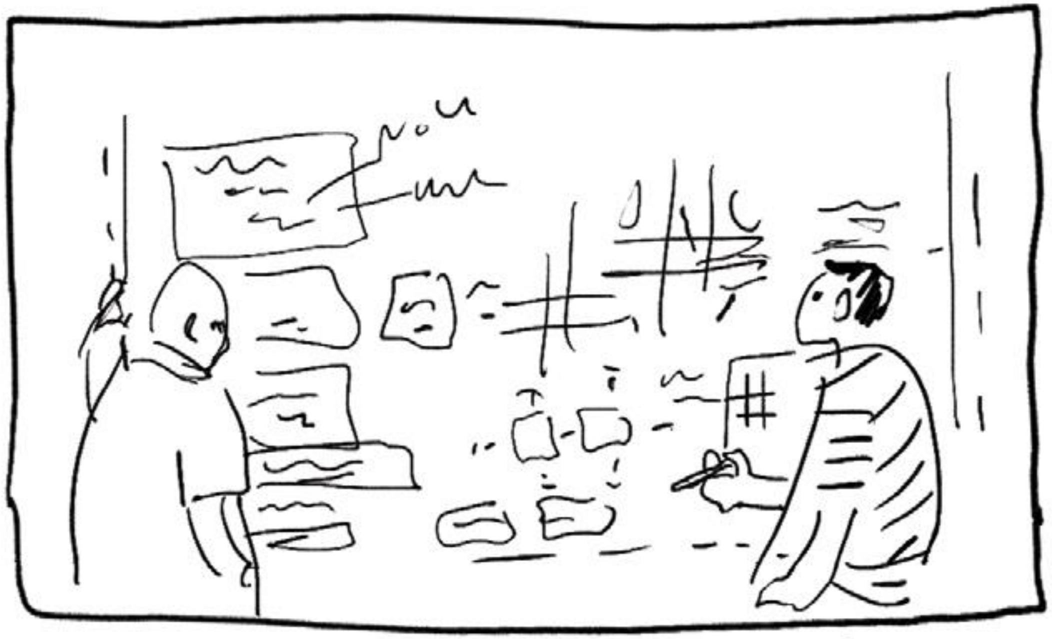
KEP #1: focus on the essence



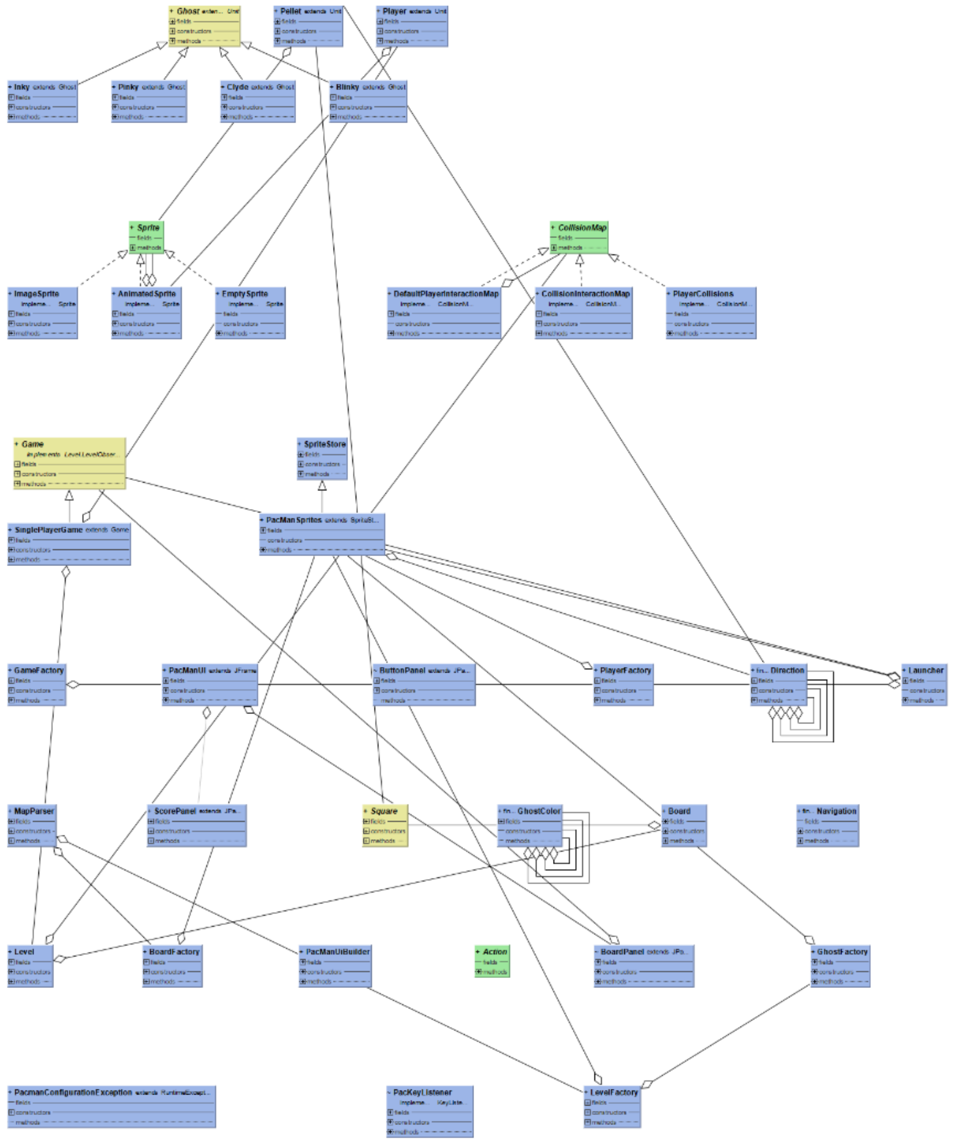
KEP #2: go as deep as needed



KEP #3: work with others



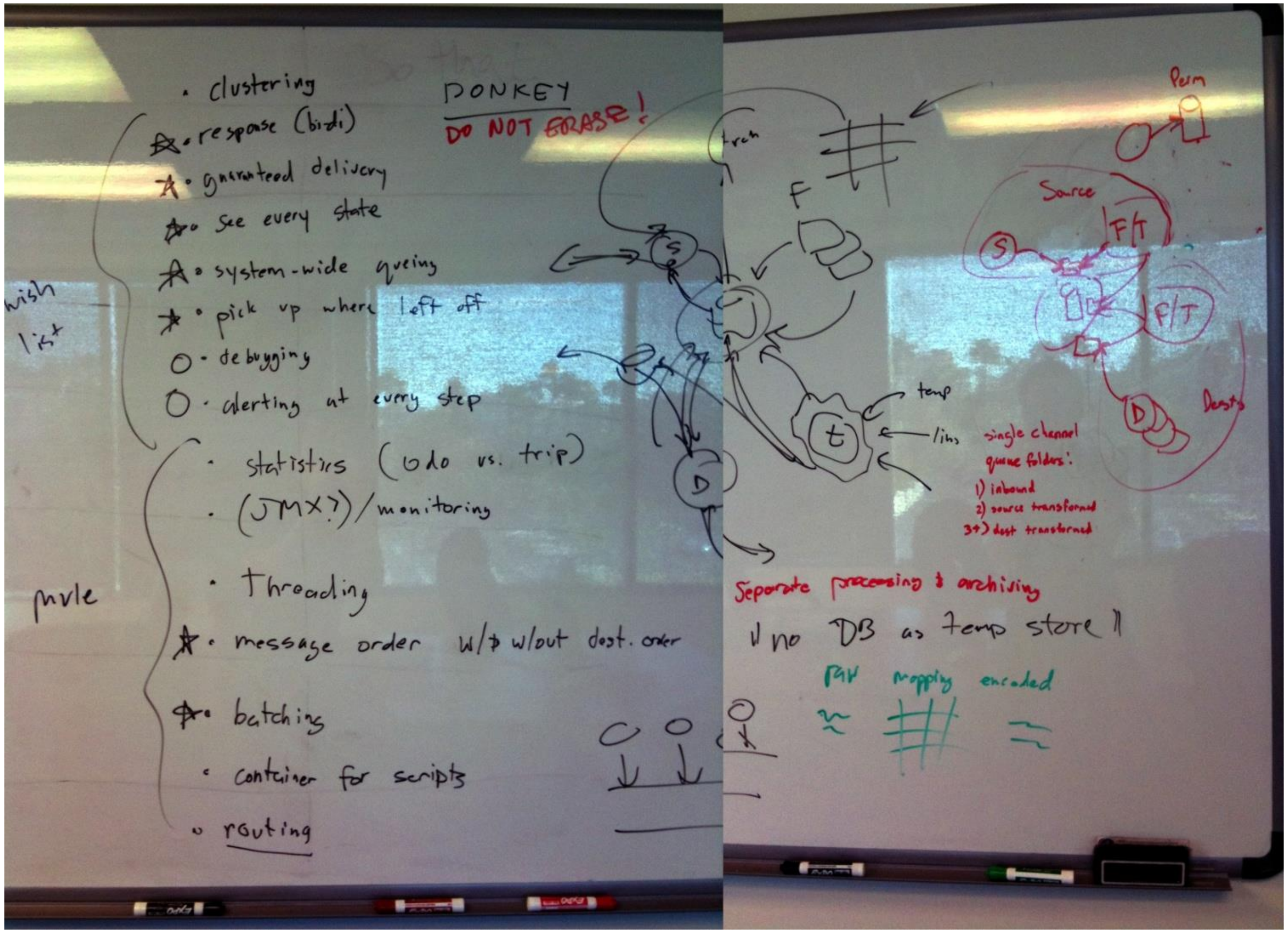
Structural versus behavioral models



Structural versus behavioral models

Structural	Behavioral
Class diagram	Use case diagram
Package diagram	Activity diagram
Component diagram	Statechart diagram
Deployment diagram	Sequence diagram
…	…

March 25, 2011 [Mirth]



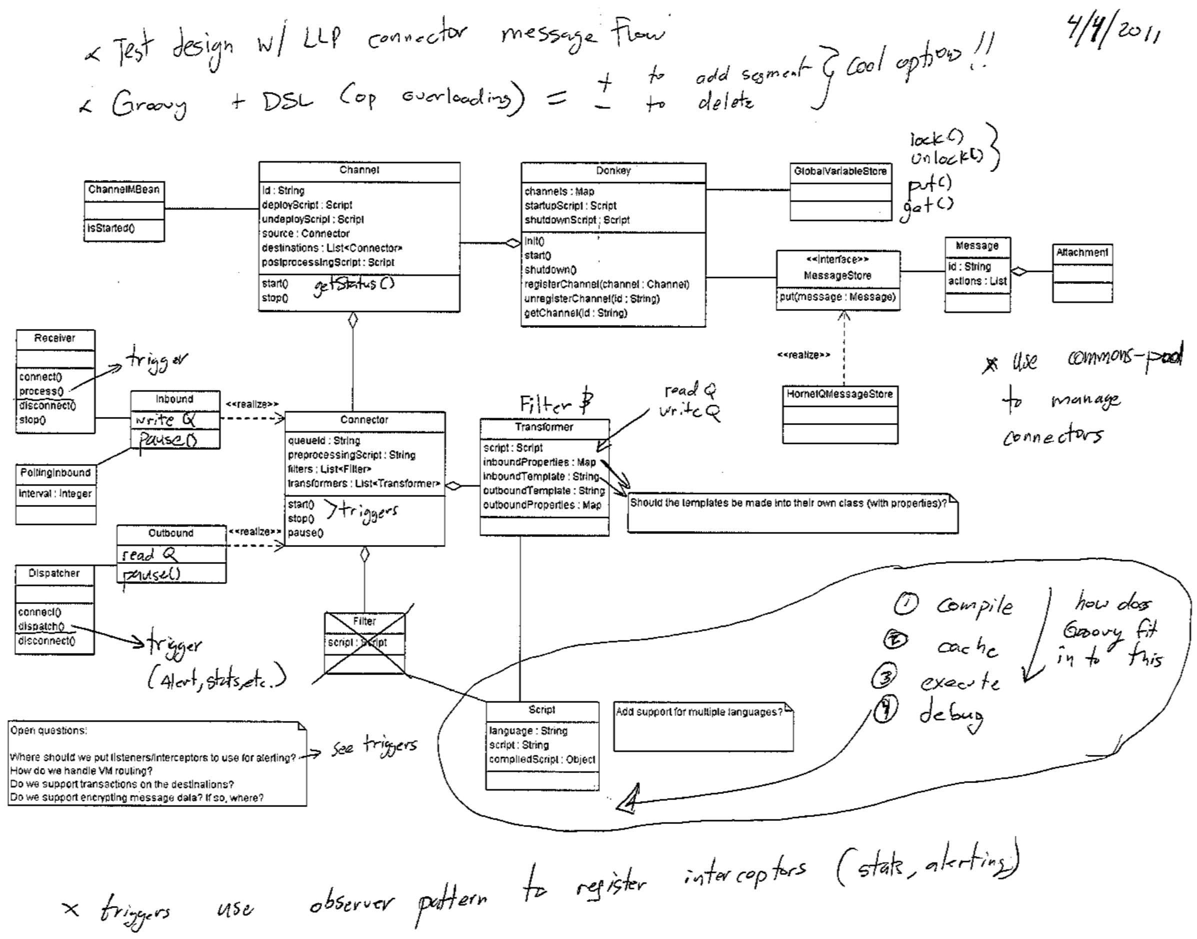
- clustering
- ☆ response (bidi)
- ☆ guaranteed delivery
- ☆ see every state
- ☆ system-wide queuing
- ☆ pick up where left off
- ○ debugging
- ○ alerting at every step
- statistics (odo vs. trip)
- (JMX?)/monitoring
- Threading
- ☆ message order w/ & w/out dest. order
- ☆ batching
- container for scripts
- routing

single channel
queue folders:
1) inbound
2) source transformed
3+) dest transformed

Separate processing & archiving

!! no DB as temp store !!

April 4, 2011



× Test design w/ LLP connector message flow

× Groovy + DSL (op overloading) = + to add segment, − to delete } cool option!!

Should the templates be made into their own class (with properties)?

Add support for multiple languages?

* use commons-pool to manage connectors

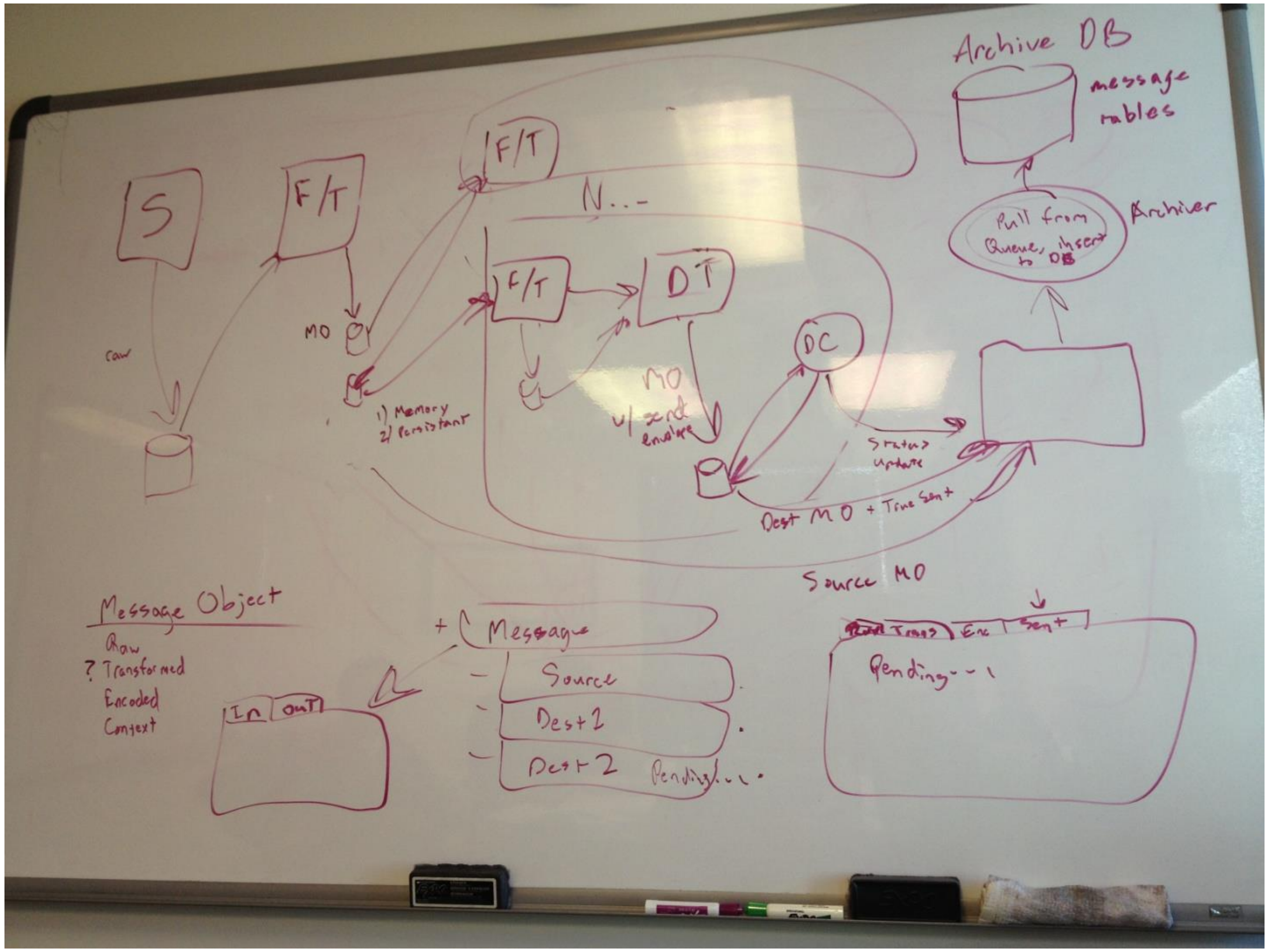
① compile ② cache ③ execute ④ debug — how does Groovy fit in to this

Open questions:

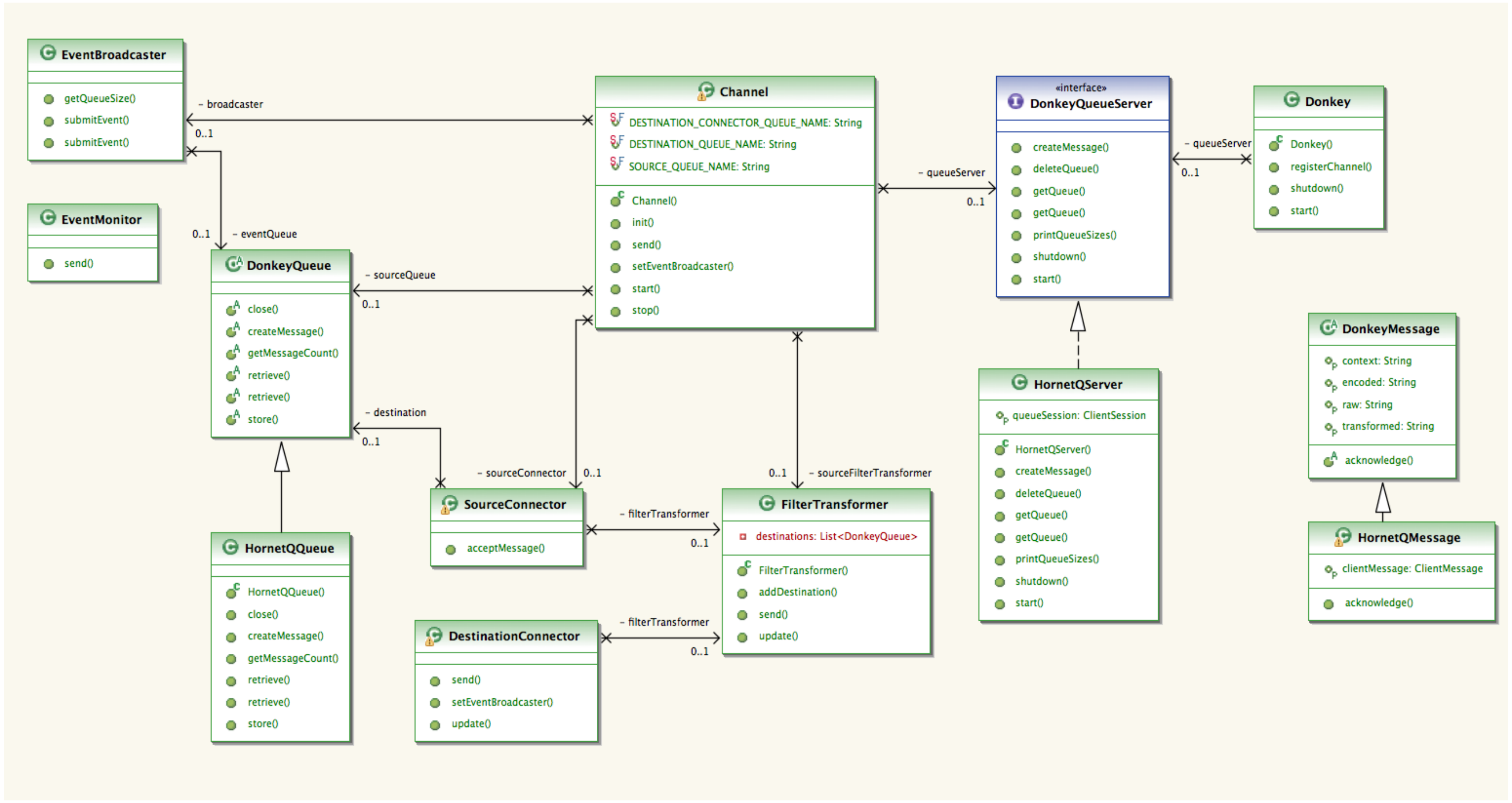
Where should we put listeners/interceptors to use for alerting?
How do we handle VM routing?
Do we support transactions on the destinations?
Do we support encrypting message data? If so, where?

× triggers use observer pattern to register interceptors (stats, alerting)

February 3, 2012

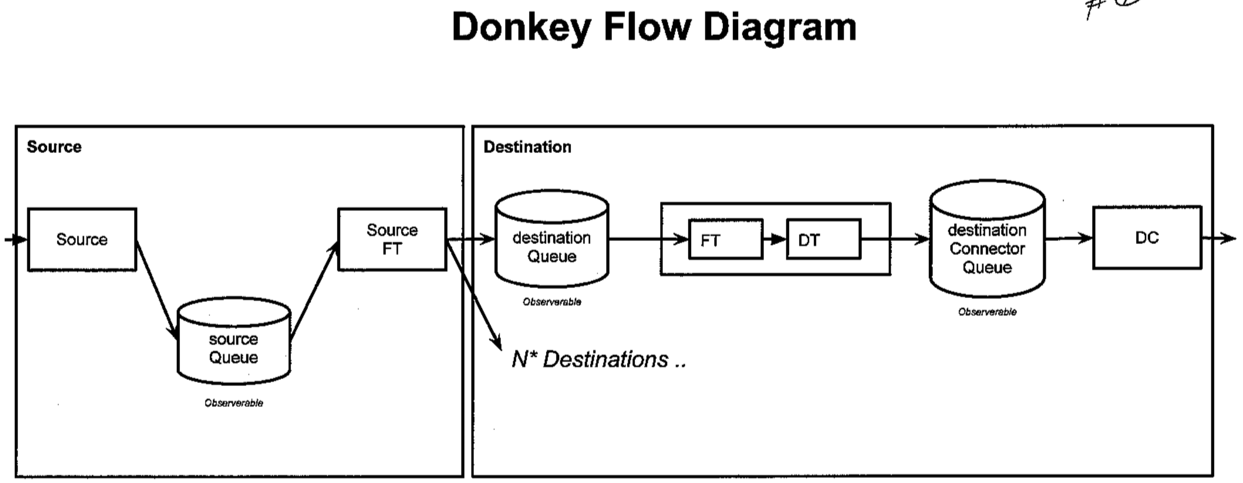


February 9, 2012

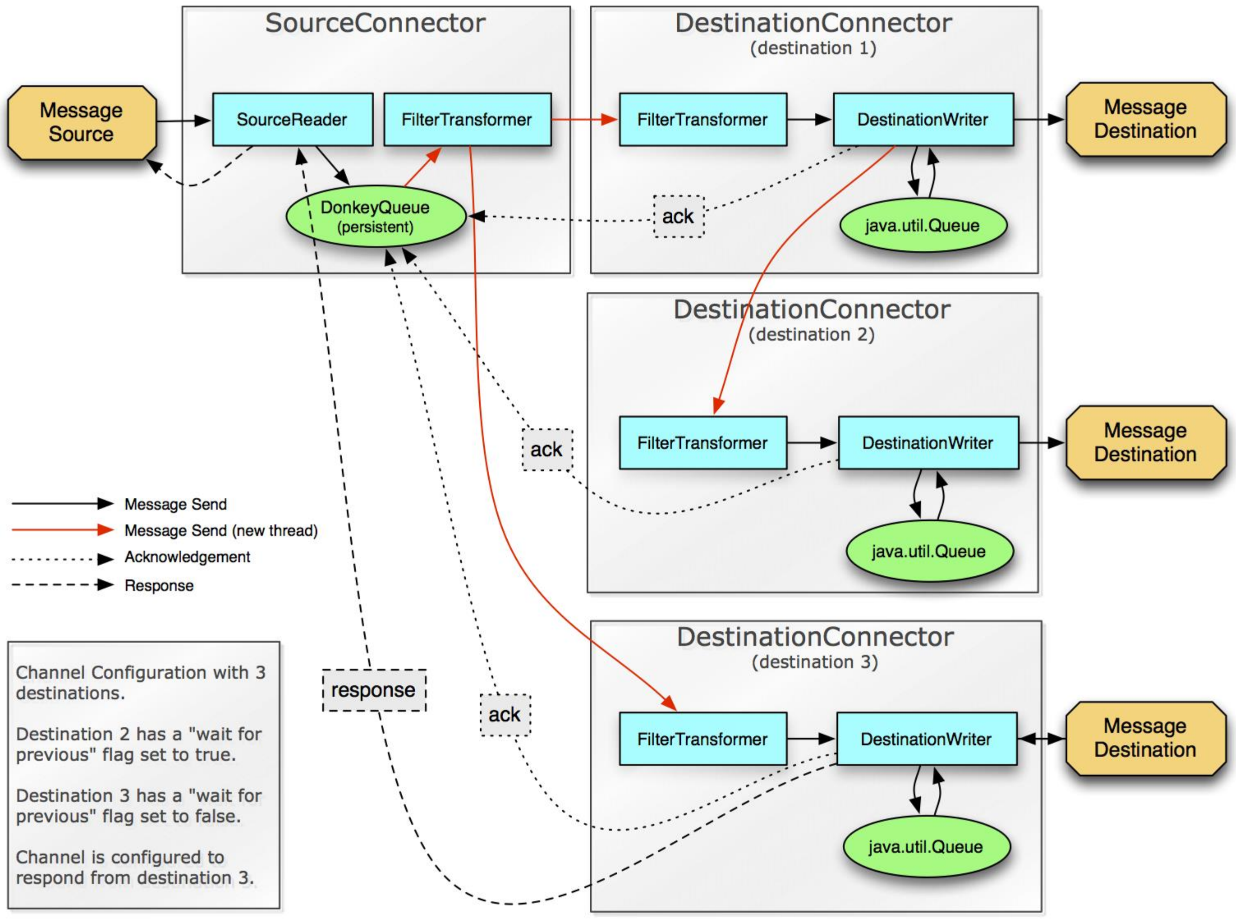


February 9, 2012

#∅



February 15, 2012



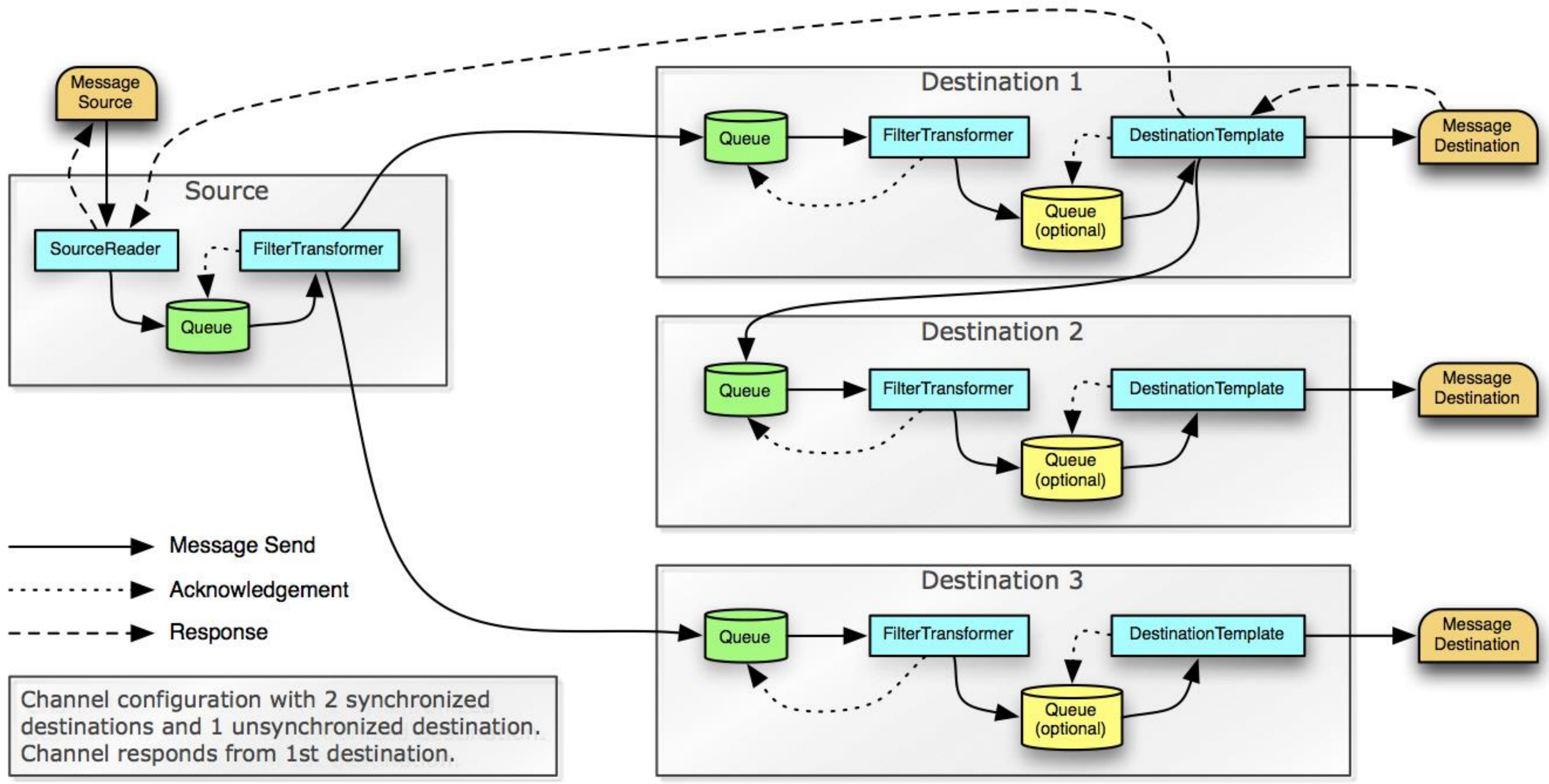
Channel Configuration with 3 destinations.

Destination 2 has a "wait for previous" flag set to true.

Destination 3 has a "wait for previous" flag set to false.

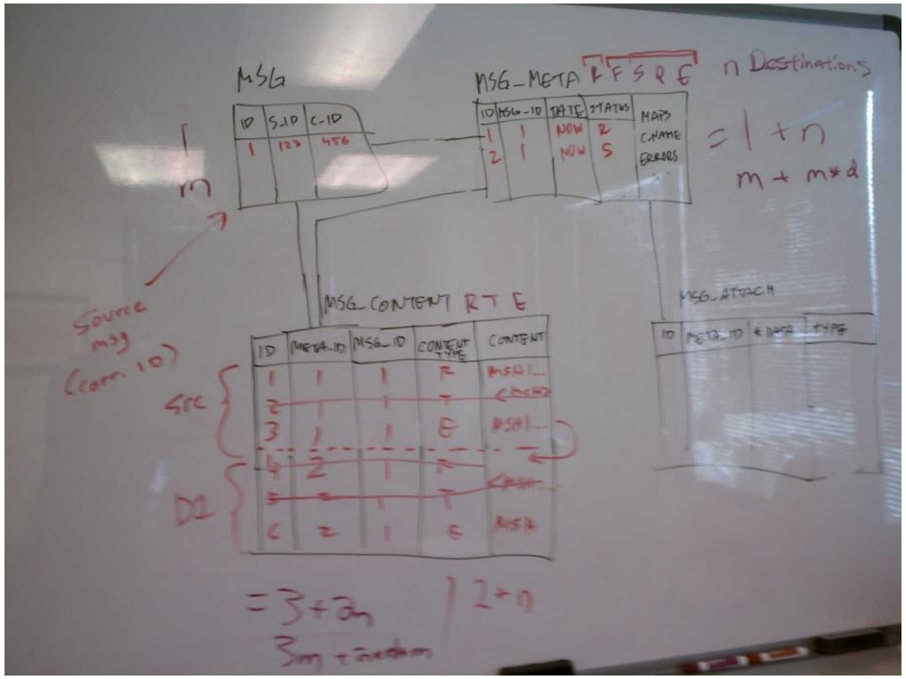
Channel is configured to respond from destination 3.

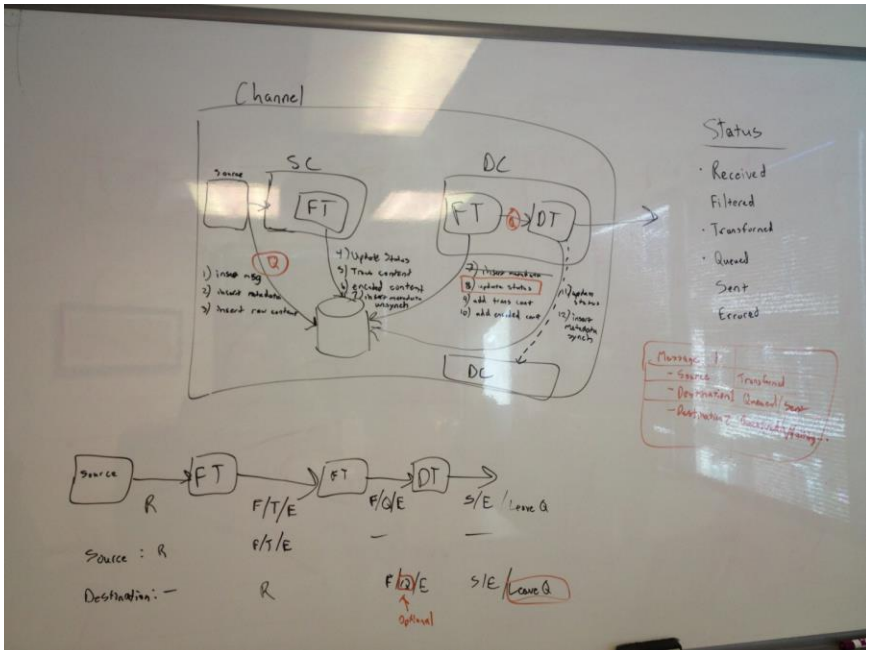
March 6, 2012



Channel configuration with 2 synchronized destinations and 1 unsynchronized destination. Channel responds from 1st destination.

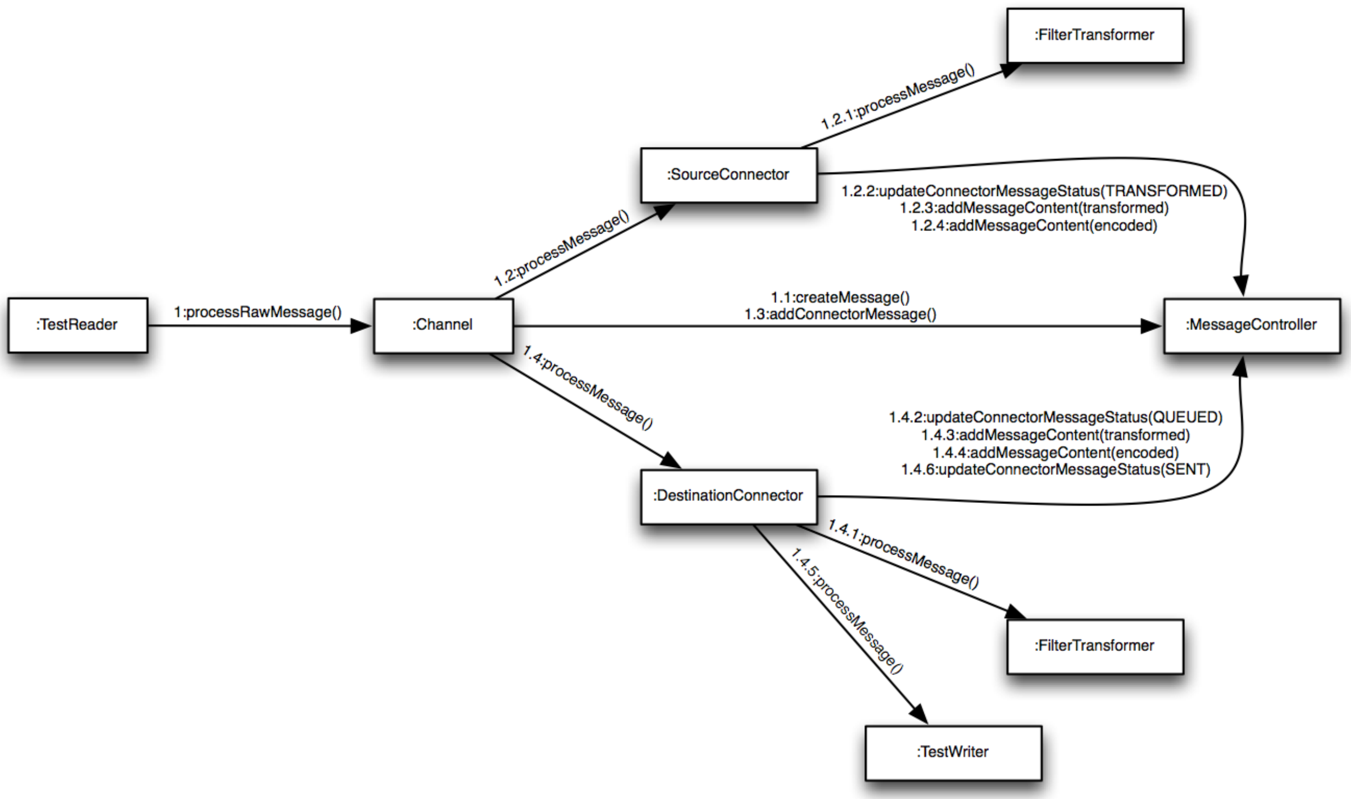
March 14, 2012





March 14, 2012

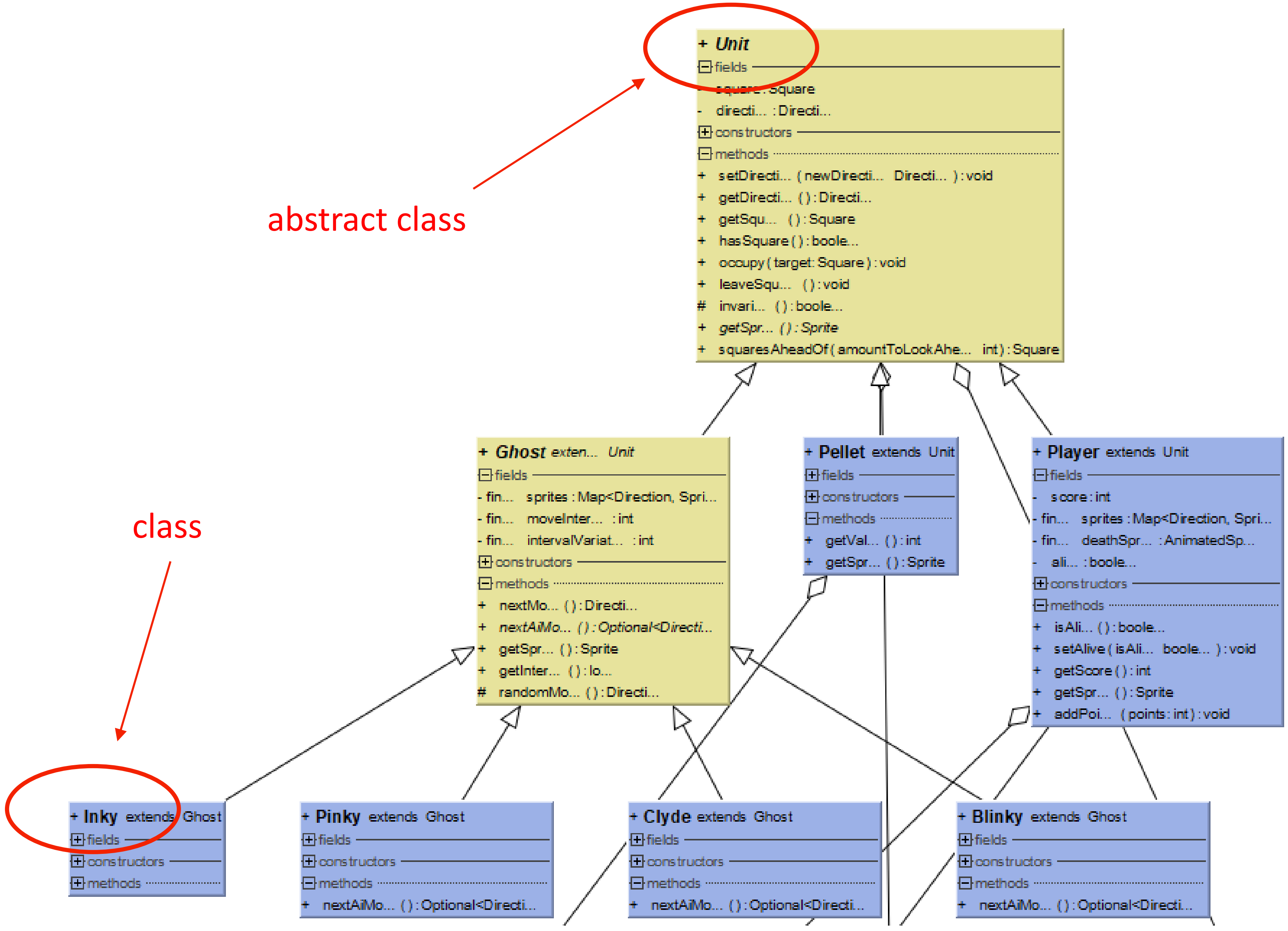
Donkey Communication Diagram


:FilterTransformer
1.2.1:processMessage()
:SourceConnector
1.2.2:updateConnectorMessageStatus(TRANSFORMED)
1.2.3:addMessageContent(transformed)
1.2.4:addMessageContent(encoded)
1.2:processMessage()
:TestReader
1:processRawMessage()
:Channel
1.1:createMessage()
1.3:addConnectorMessage()
:MessageController
1.4:processMessage()
1.4.2:updateConnectorMessageStatus(QUEUED)
1.4.3:addMessageContent(transformed)
1.4.4:addMessageContent(encoded)
1.4.6:updateConnectorMessageStatus(SENT)
:DestinationConnector
1.4.1:processMessage()
1.4.5:processMessage()
:FilterTransformer
:TestWriter

Morals of the story

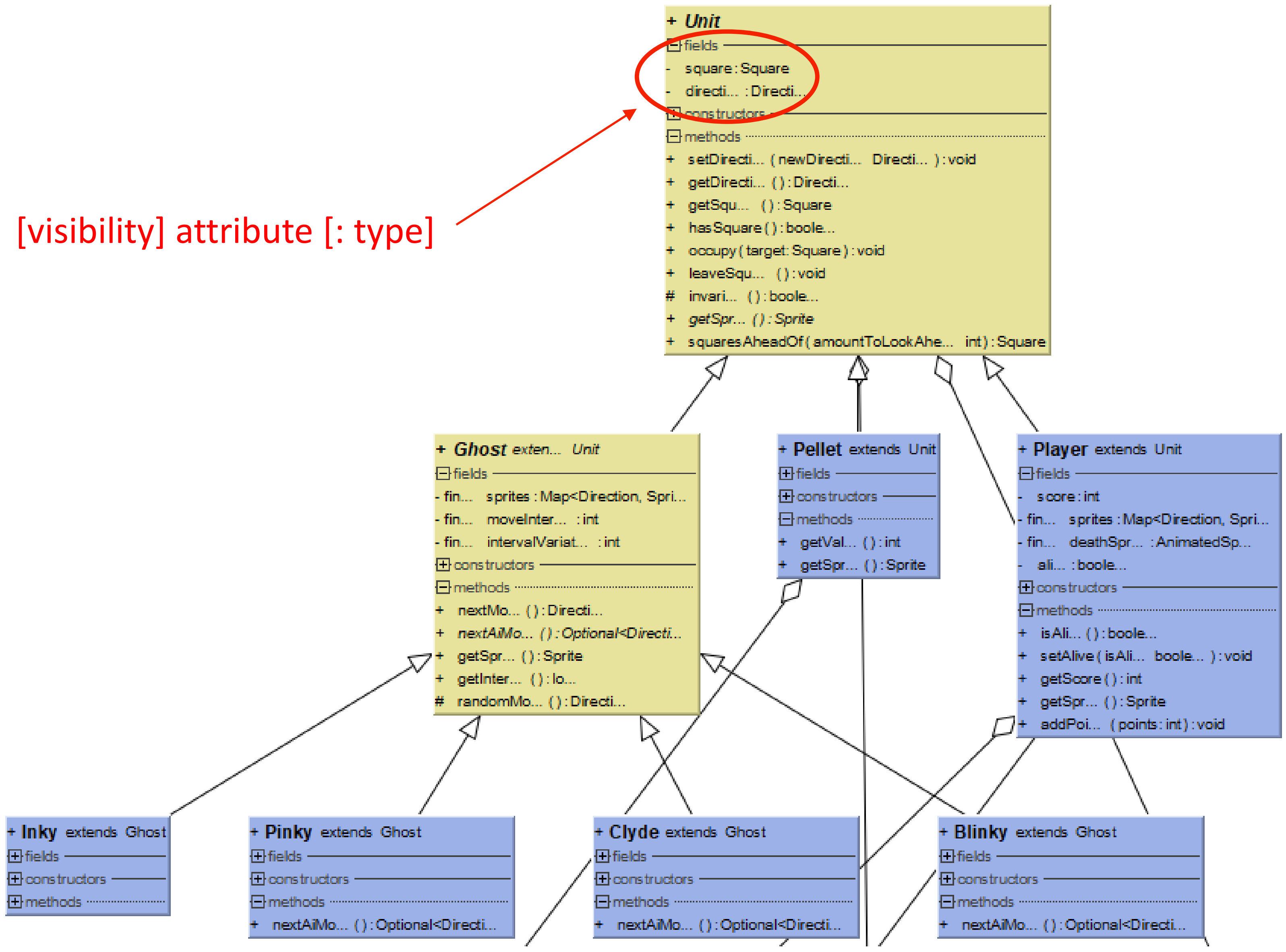
- Understanding what your current code really, really does is essential
- Modeling matters in the real world
- Structural and behavioral models are used
- Not all models are formal and precise

UML class diagram notation


abstract class
class
+ Unit
fields
- square : Square
- directi... : Directi...
constructors
methods
+ setDirecti... (newDirecti... Directi...) : void
+ getDirecti... ( ) : Directi...
+ getSqu... ( ) : Square
+ hasSquare ( ) : boole...
+ occupy (target: Square) : void
+ leaveSqu... ( ) : void
# invari... ( ) : boole...
+ getSpr... ( ) : Sprite
+ squaresAheadOf (amountToLookAhe... int) : Square
+ Ghost exten... Unit
fields
- fin... sprites : Map<Direction, Spri...
- fin... moveInter... : int
- fin... intervalVariat... : int
constructors
methods
+ nextMo... ( ) : Directi...
+ nextAiMo... ( ) : Optional<Directi...
+ getSpr... ( ) : Sprite
+ getInter... ( ) : lo...
# randomMo... ( ) : Directi...
+ Pellet extends Unit
fields
constructors
methods
+ getVal... ( ) : int
+ getSpr... ( ) : Sprite
+ Player extends Unit
fields
- score : int
- fin... sprites : Map<Direction, Spri...
- fin... deathSpr... : AnimatedSp...
- ali... : boole...
constructors
methods
+ isAli... ( ) : boole...
+ setAlive (isAli... boole...) : void
+ getScore ( ) : int
+ getSpr... ( ) : Sprite
+ addPoi... (points : int) : void
+ Inky extends Ghost
fields
constructors
methods
+ Pinky extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...
+ Clyde extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...
+ Blinky extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...

UML class diagram notation

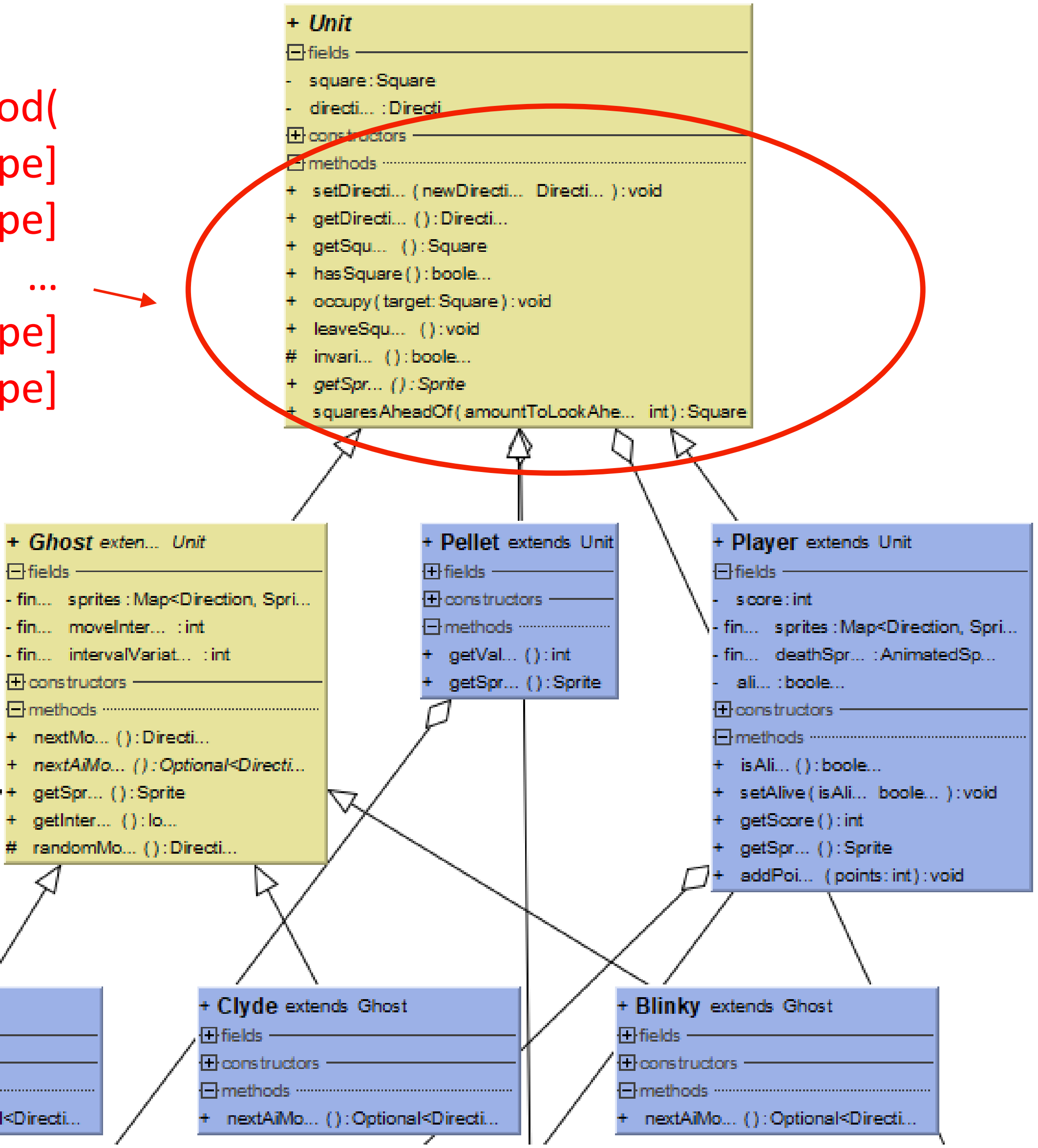
[visibility] attribute [: type]



+ Unit
fields
- square : Square
- directi... : Directi...
constructors
methods
+ setDirecti... (newDirecti... Directi...) : void
+ getDirecti... ( ) : Directi...
+ getSqu... ( ) : Square
+ hasSquare ( ) : boole...
+ occupy (target: Square) : void
+ leaveSqu... ( ) : void
# invari... ( ) : boole...
+ getSpr... ( ) : Sprite
+ squaresAheadOf (amountToLookAhe... int) : Square
+ Ghost exten... Unit
fields
- fin... sprites : Map<Direction, Spri...
- fin... moveInter... : int
- fin... intervalVariat... : int
constructors
methods
+ nextMo... ( ) : Directi...
+ nextAiMo... ( ) : Optional<Directi...
+ getSpr... ( ) : Sprite
+ getInter... ( ) : lo...
# randomMo... ( ) : Directi...
+ Pellet extends Unit
fields
constructors
methods
+ getVal... ( ) : int
+ getSpr... ( ) : Sprite
+ Player extends Unit
fields
- score : int
- fin... sprites : Map<Direction, Spri...
- fin... deathSpr... : AnimatedSp...
- ali... : boole...
constructors
methods
+ isAli... ( ) : boole...
+ setAlive (isAli... boole...) : void
+ getScore ( ) : int
+ getSpr... ( ) : Sprite
+ addPoi... (points : int) : void
+ Inky extends Ghost
fields
constructors
methods
+ Pinky extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...
+ Clyde extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...
+ Blinky extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...

UML class diagram notation

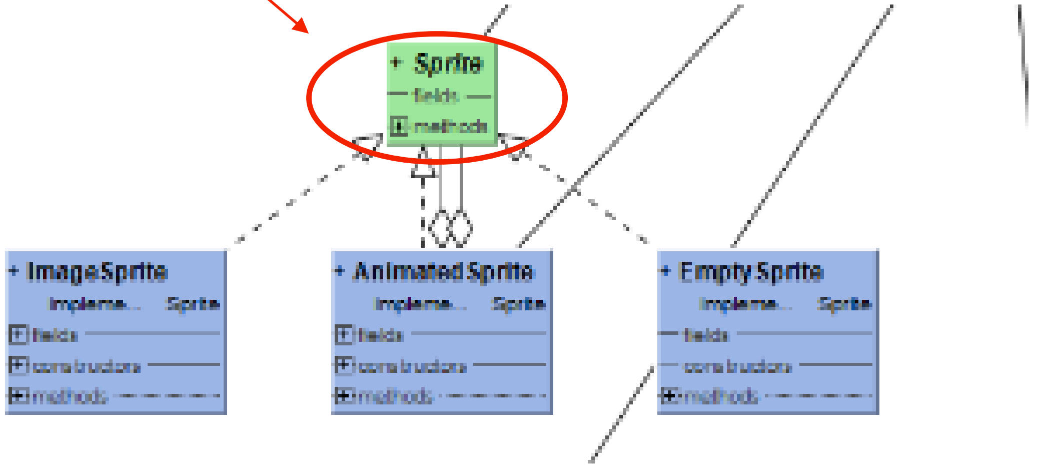
[visibility] method(
[direction][parameter1][: type]
, [direction][parameter2][: type]
…
, [direction][parametern][: type]
)[: return type]



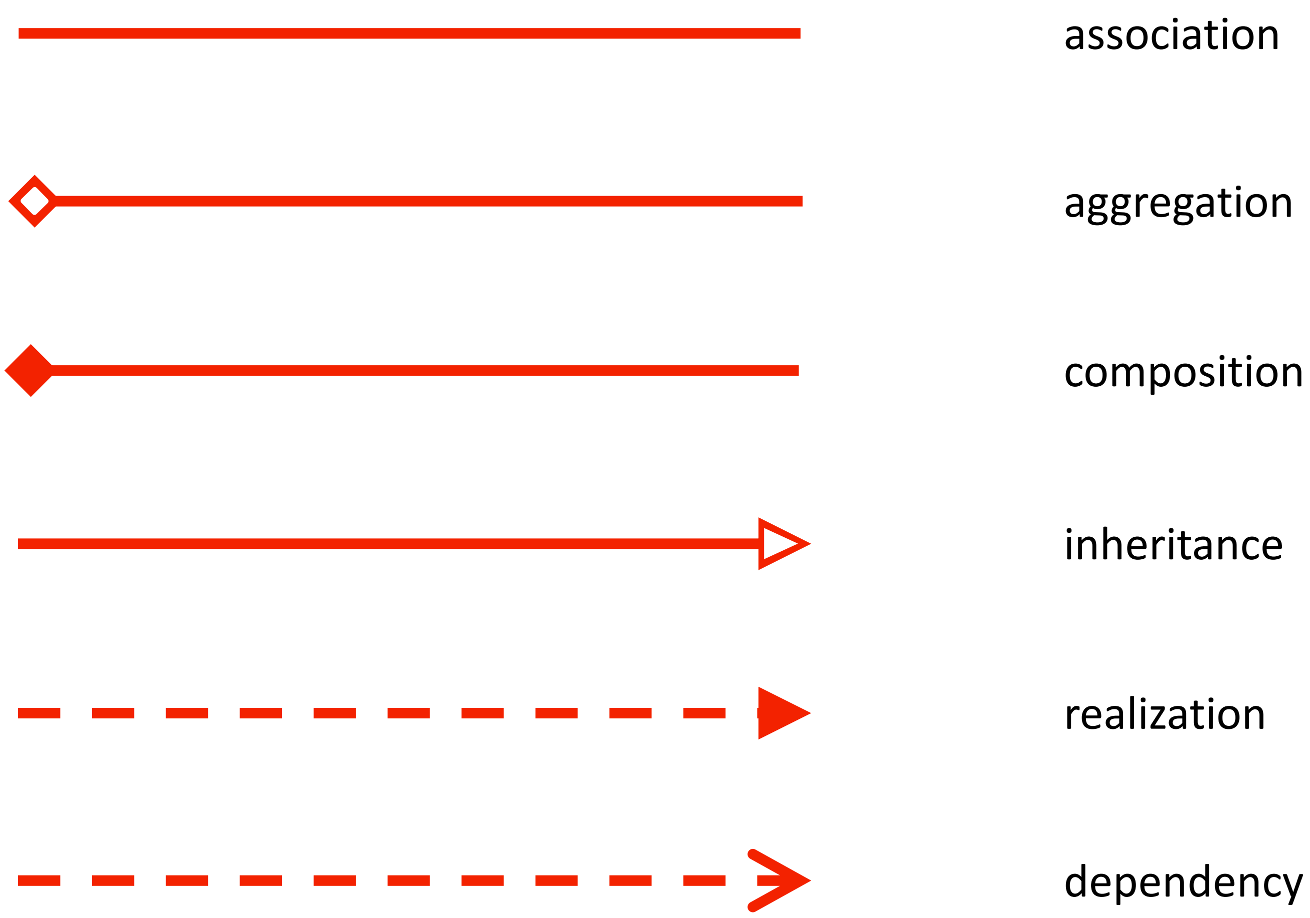
+ Unit
fields
- square : Square
- directi... : Directi
constructors
methods
+ setDirecti... (newDirecti... Directi...) : void
+ getDirecti... ( ) : Directi...
+ getSqu... ( ) : Square
+ hasSquare ( ) : boole...
+ occupy (target: Square) : void
+ leaveSqu... ( ) : void
# invari... ( ) : boole...
+ getSpr... ( ) : Sprite
+ squaresAheadOf (amountToLookAhe... int) : Square
+ Ghost exten... Unit
fields
- fin... sprites : Map<Direction, Spri...
- fin... moveInter... : int
- fin... intervalVariat... : int
constructors
methods
+ nextMo... ( ) : Directi...
+ nextAiMo... ( ) : Optional<Directi...
+ getSpr... ( ) : Sprite
+ getInter... ( ) : lo...
# randomMo... ( ) : Directi...
+ Pellet extends Unit
fields
constructors
methods
+ getVal... ( ) : int
+ getSpr... ( ) : Sprite
+ Player extends Unit
fields
- score : int
- fin... sprites : Map<Direction, Spri...
- fin... deathSpr... : AnimatedSp...
- ali... : boole...
constructors
methods
+ isAli... ( ) : boole...
+ setAlive (isAli... boole...) : void
+ getScore ( ) : int
+ getSpr... ( ) : Sprite
+ addPoi... (points : int) : void
+ Inky extends Ghost
fields
constructors
methods
+ Pinky extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...
+ Clyde extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...
+ Blinky extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...

UML class diagram notation

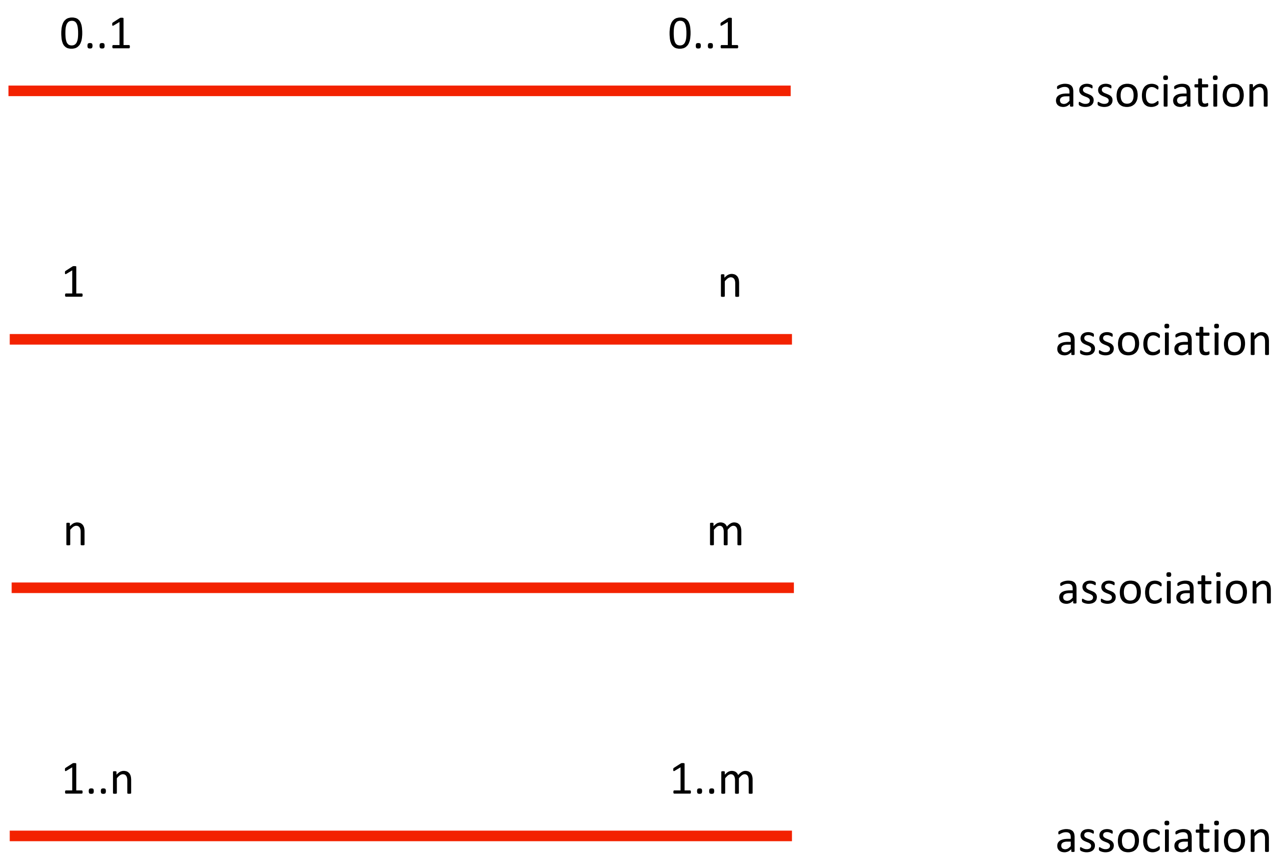
interface



UML class diagram notations

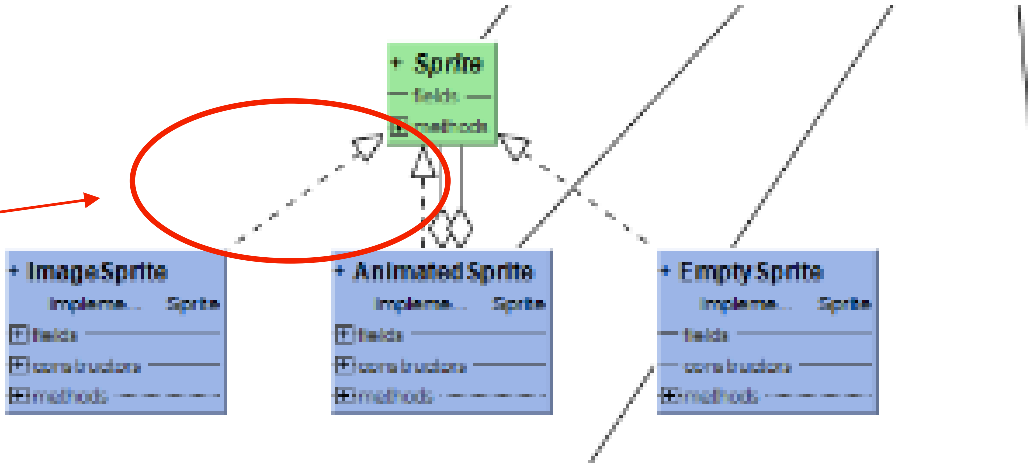


UML class diagram notations

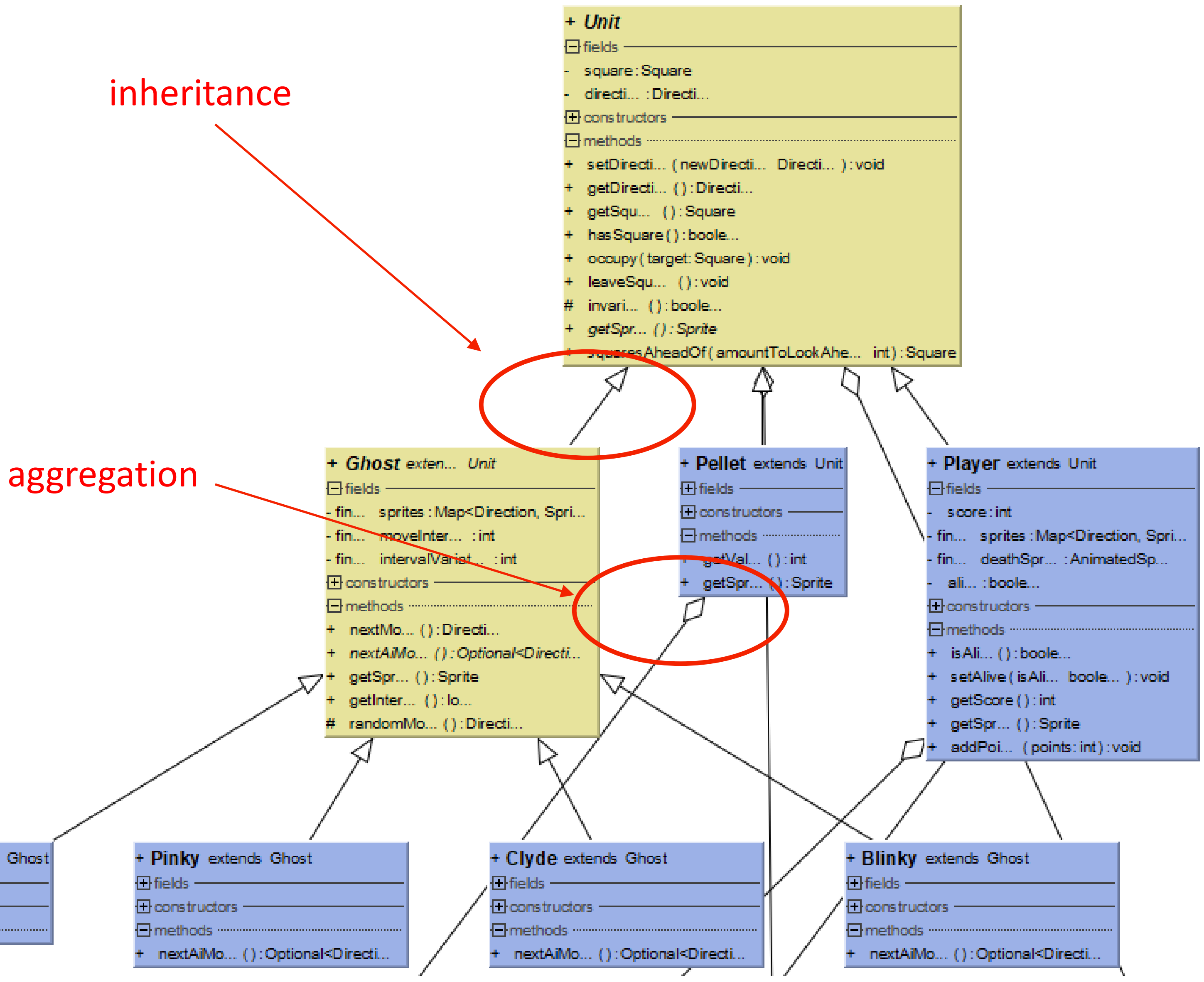


UML class diagram notation

realizes relation

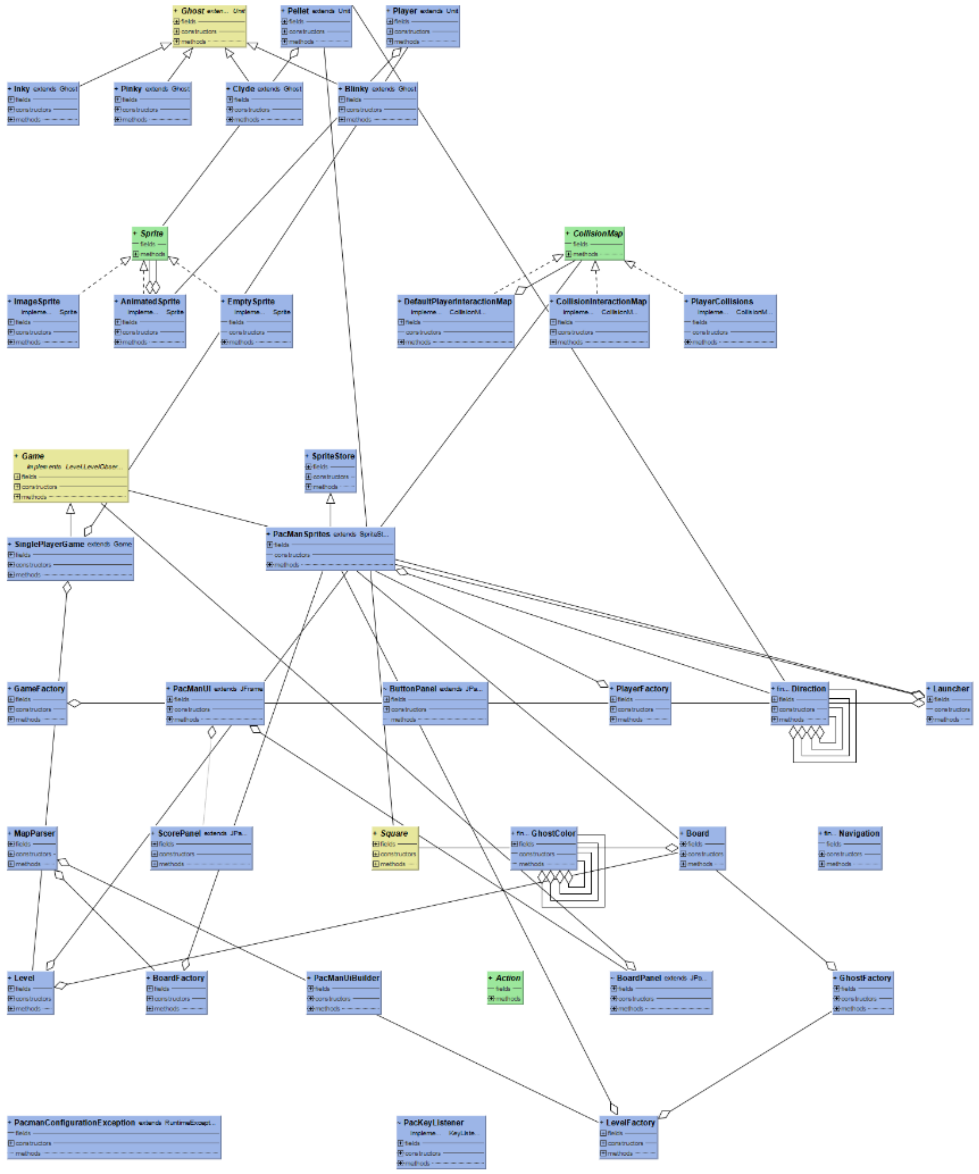


UML class diagram notation

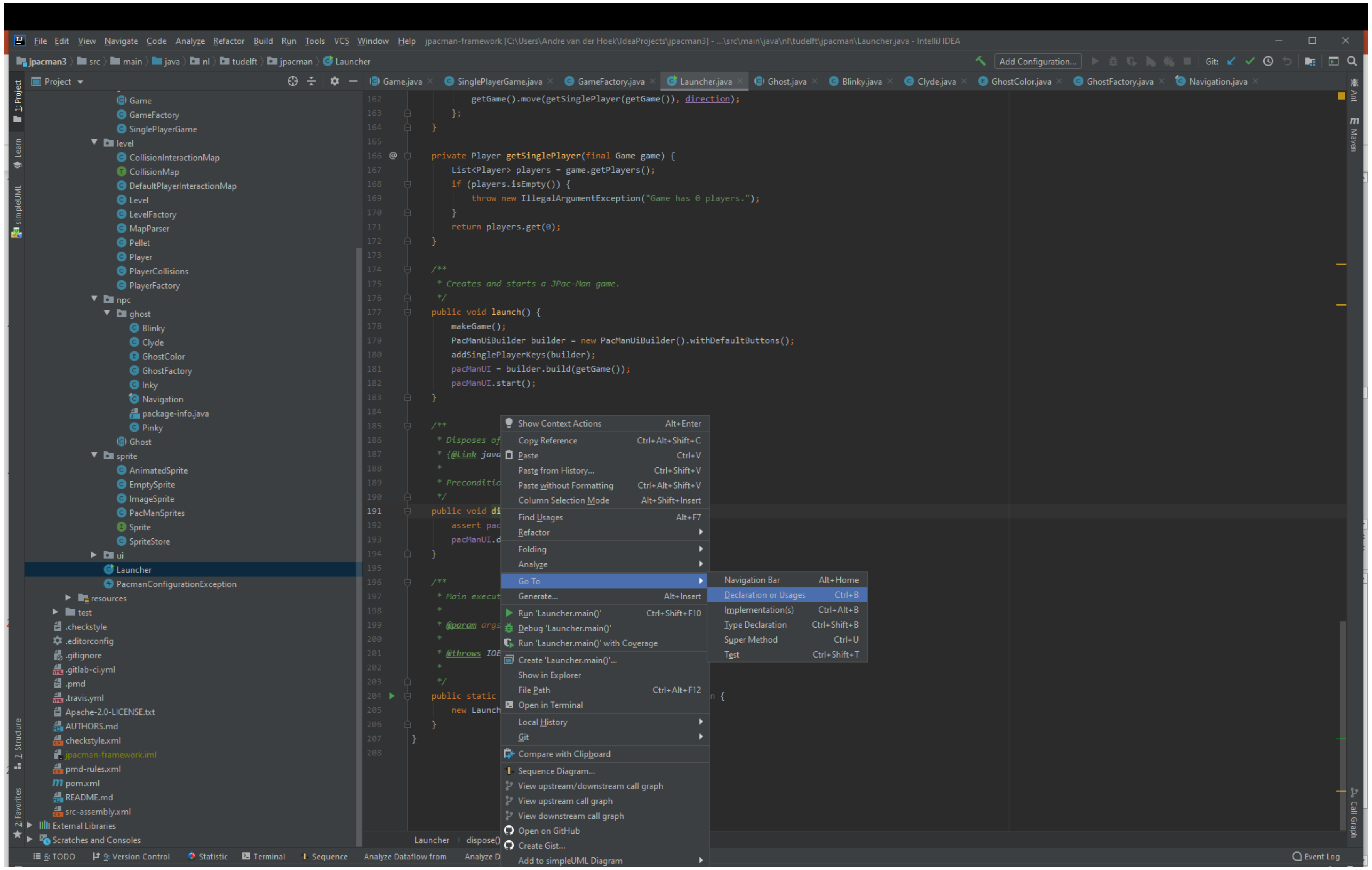

inheritance
aggregation
+ Unit
fields
- square : Square
- directi... : Directi...
constructors
methods
+ setDirecti... (newDirecti... Directi...) : void
+ getDirecti... ( ) : Directi...
+ getSqu... ( ) : Square
+ hasSquare ( ) : boole...
+ occupy (target: Square) : void
+ leaveSqu... ( ) : void
# invari... ( ) : boole...
+ getSpr... ( ) : Sprite
+ squaresAheadOf (amountToLookAhe... int) : Square
+ Ghost exten... Unit
fields
- fin... sprites : Map<Direction, Spri...
- fin... moveInter... : int
- fin... intervalVariat... : int
constructors
methods
+ nextMo... ( ) : Directi...
+ nextAiMo... ( ) : Optional<Directi...
+ getSpr... ( ) : Sprite
+ getInter... ( ) : lo...
# randomMo... ( ) : Directi...
+ Pellet extends Unit
fields
constructors
methods
+ getVal... ( ) : int
+ getSpr... ( ) : Sprite
+ Player extends Unit
fields
- score : int
- fin... sprites : Map<Direction, Spri...
- fin... deathSpr... : AnimatedSp...
- ali... : boole...
constructors
methods
+ isAli... ( ) : boole...
+ setAlive (isAli... boole...) : void
+ getScore ( ) : int
+ getSpr... ( ) : Sprite
+ addPoi... (points : int) : void
+ Inky extends Ghost
fields
constructors
methods
+ Pinky extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...
+ Clyde extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...
+ Blinky extends Ghost
fields
constructors
methods
+ nextAiMo... ( ) : Optional<Directi...

Break

UML class diagram

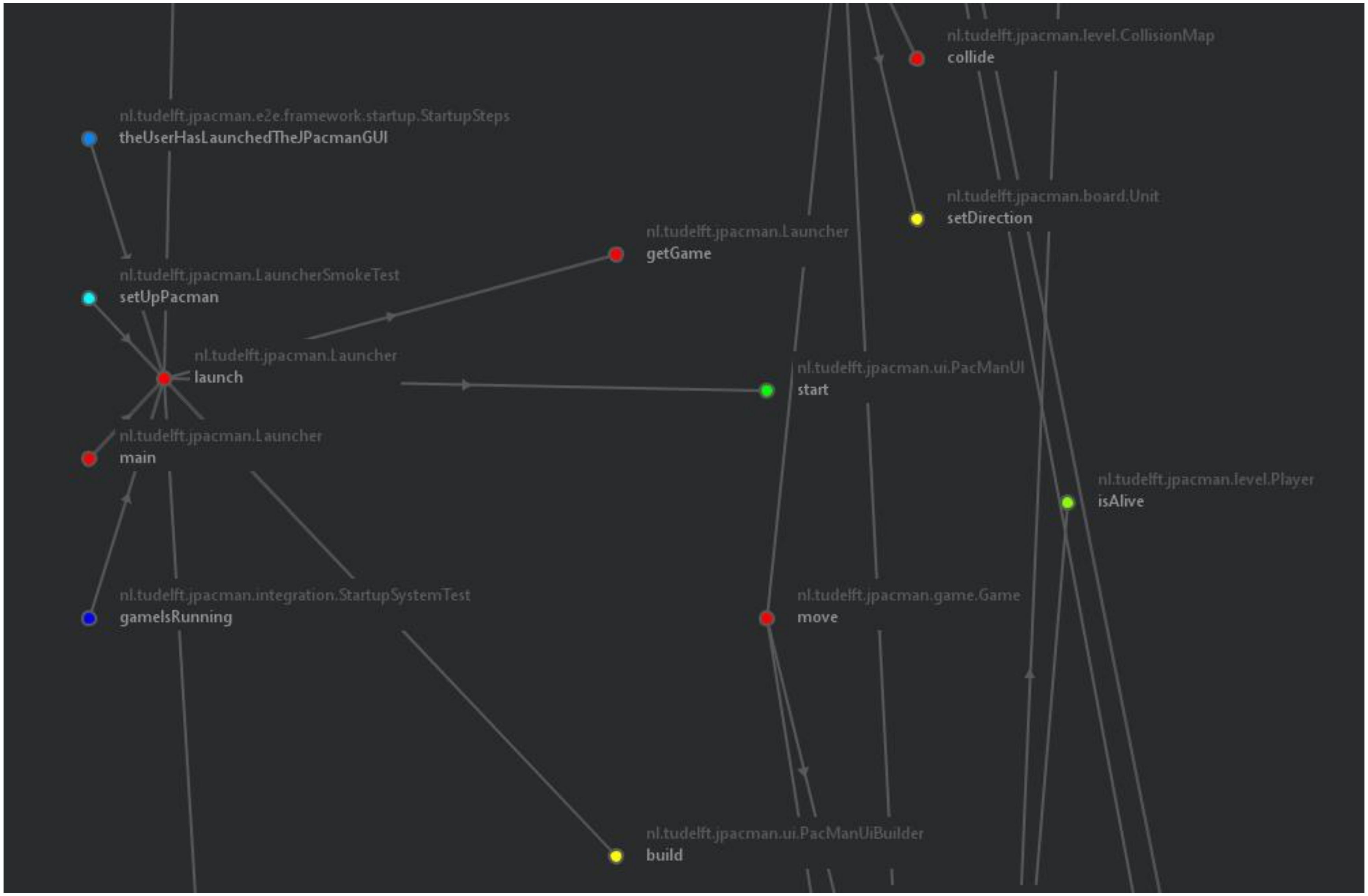


Following the trail

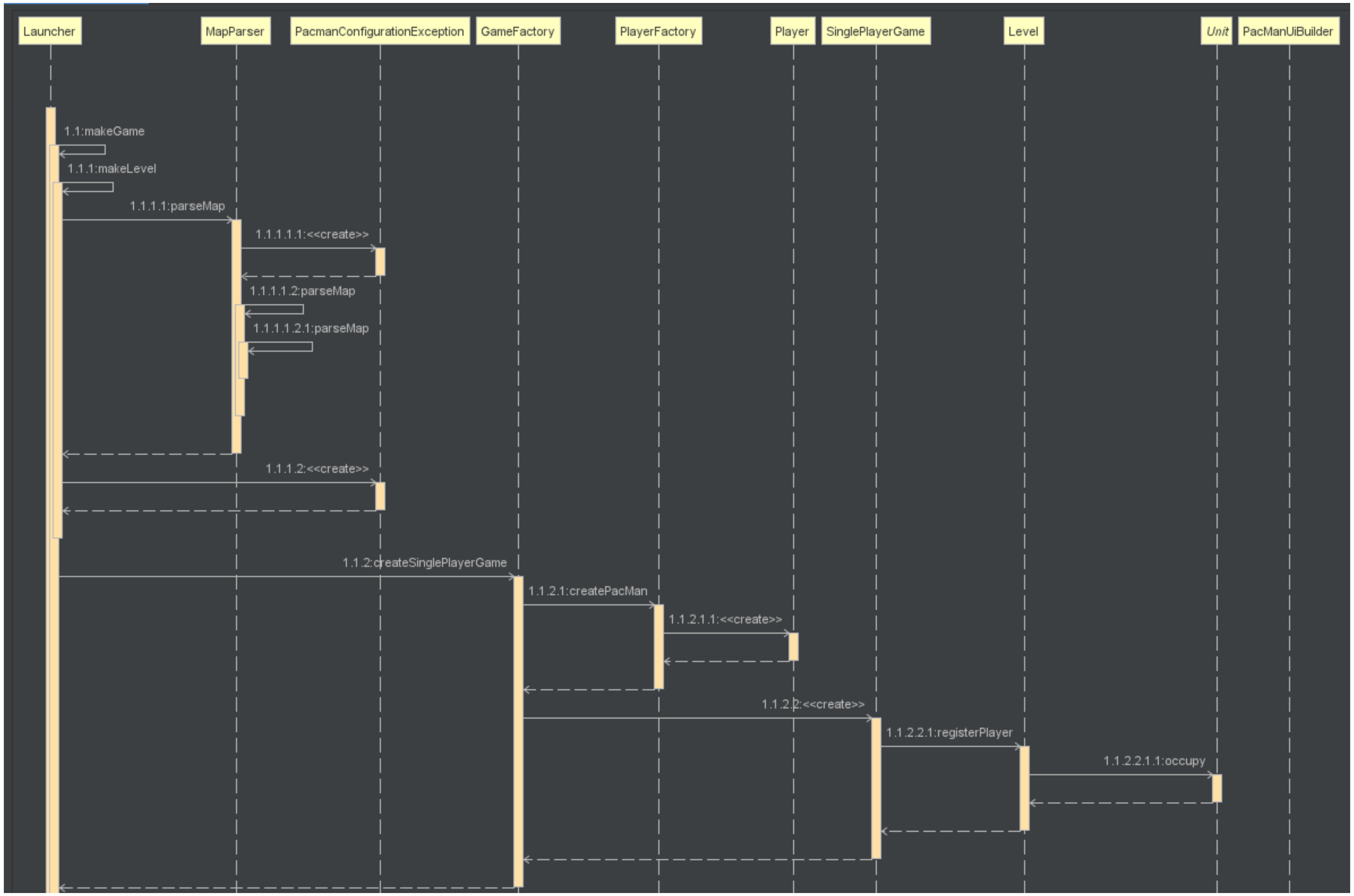


```
            getGame().move(getSinglePlayer(getGame()), direction);
        };
    }

    private Player getSinglePlayer(final Game game) {
        List<Player> players = game.getPlayers();
        if (players.isEmpty()) {
            throw new IllegalArgumentException("Game has 0 players.");
        }
        return players.get(0);
    }

    /**
     * Creates and starts a JPac-Man game.
     */
    public void launch() {
        makeGame();
        PacManUiBuilder builder = new PacManUiBuilder().withDefaultButtons();
        addSinglePlayerKeys(builder);
        pacManUI = builder.build(getGame());
        pacManUI.start();
    }

    /**
     * Disposes of
     * {@link java
     *
     * Preconditio
     */
    public void di
        assert pac
        pacManUI.d
    }

    /**
     * Main execut
     *
     * @param args
     *
     * @throws IOE
     *
     */
    public static                          n {
        new Launch
    }
}
```

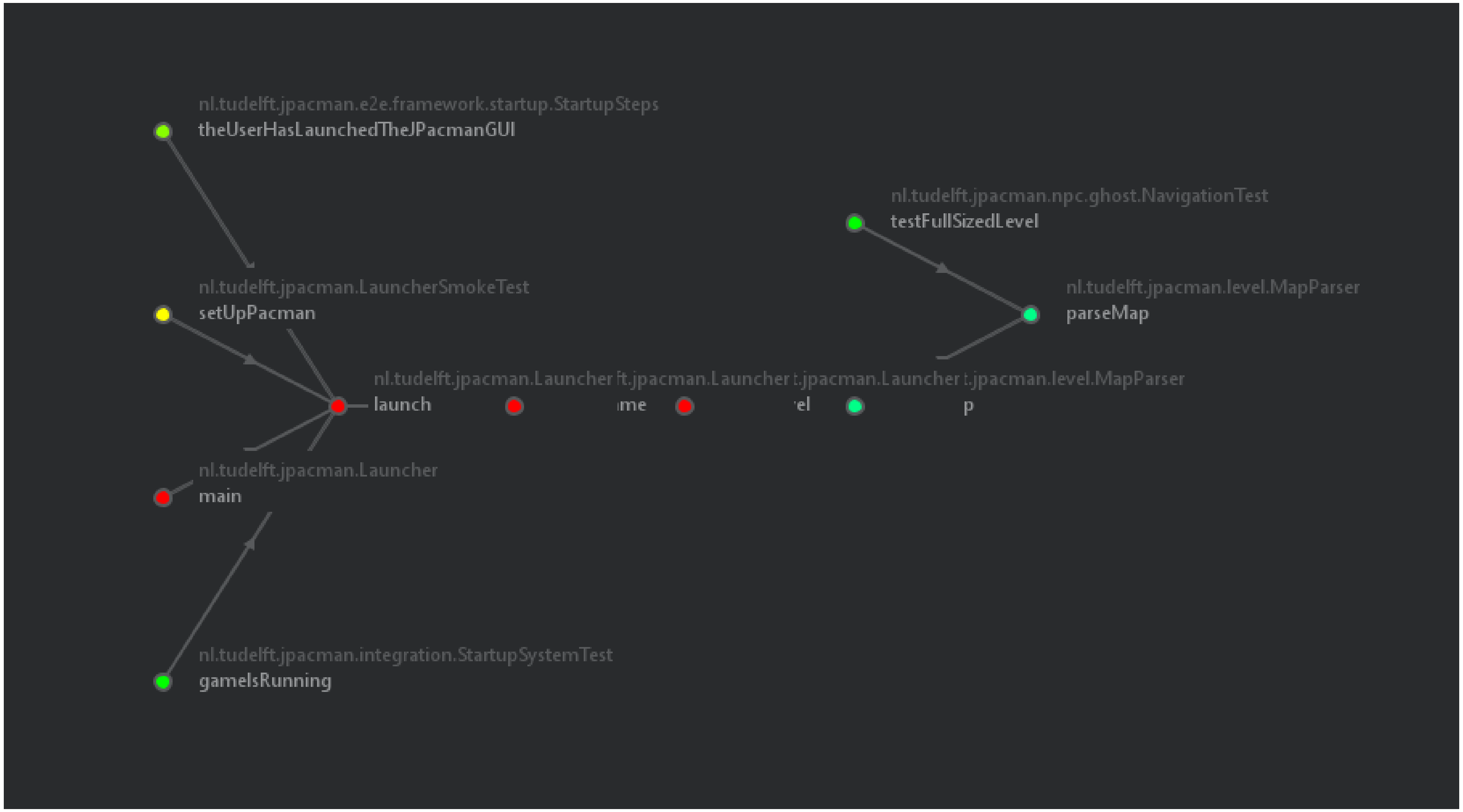
Call graphs


nl.tudelft.jpacman.level.CollisionMap
collide
nl.tudelft.jpacman.e2e.framework.startup.StartupSteps
theUserHasLaunchedTheJPacmanGUI
nl.tudelft.jpacman.board.Unit
setDirection
nl.tudelft.jpacman.Launcher
getGame
nl.tudelft.jpacman.LauncherSmokeTest
setUpPacman
nl.tudelft.jpacman.Launcher
launch
nl.tudelft.jpacman.ui.PacManUI
start
nl.tudelft.jpacman.Launcher
main
nl.tudelft.jpacman.level.Player
isAlive
nl.tudelft.jpacman.integration.StartupSystemTest
gameIsRunning
nl.tudelft.jpacman.game.Game
move
nl.tudelft.jpacman.ui.PacManUiBuilder
build

UML sequence diagram



Going backwards (call graphs)


nl.tudelft.jpacman.e2e.framework.startup.StartupSteps
theUserHasLaunchedTheJPacmanGUI
nl.tudelft.jpacman.npc.ghost.NavigationTest
testFullSizedLevel
nl.tudelft.jpacman.LauncherSmokeTest
setUpPacman
nl.tudelft.jpacman.level.MapParser
parseMap
nl.tudelft.jpacman.Launcher
launch
ft.jpacman.Launcher
ıme
t.jpacman.Launcher
'el
t.jpacman.level.MapParser
p
nl.tudelft.jpacman.Launcher
main
nl.tudelft.jpacman.integration.StartupSystemTest
gameIsRunning

Important frequent question #2 and #3

What parts of the code rely on this feature (or class, or method, or variable, or…)?

What parts of the code does this feature (or class, or method, or variable, or…) rely on?

Homework (team)

- With your team, decide upon two features that are **essential** in your system, and imagine that each of the two features will need to undergo some kind of change to be implemented by someone else

- Prepare a packet, per feature, that would assist that other person in understanding where the feature is located, what other parts of the system may be relevant, and how those parts are relevant

Homework (team)

- Due date: start of class next week
- Submit via a GitHub pull request that creates a homework_2 folder in your team's folder, as many files as you need, with the following naming convention:
  - hw2_<team_name>_<filename>.<extension>
- Bring a printed copy of your diagrams
- Start early

Homework (individual)

- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/
- https://online.visual-paradigm.com/diagrams/tutorials/sequence-diagram-tutorial/
- https://www.youtube.com/watch?v=UI6lqHOVHic
- https://www.youtube.com/watch?v=pCK6prSq8aw

Homework (individual)

- Make sure to regularly update your personal diary, including an entry for today's lecture

Optional advanced material

- Study design structure matrices

And now…

- …welcome Consuelo Lopez!