DOMAIN MODEL
/////////////////////////////////

Tile

Board

Carcassonne
turn
stack

Segment

Player
score

Fields

Cities
score
shields

Roads
score

Monasteries
score

Features
completed
ownership

Meeple

<- place

<- score

<- occupy

1
1..72
1
1
1
1
1
5
2..5
1
7
1..*
0..*
1
0...*

Ledgend:
->   direction of action

Player

Carcassonne

start(#players, playerID)

For each turn until stak empty

a newly drawn tile

until valid

placeTile (tile, location, orientation)

invalid

until valid

placeMeeple (feature)

invalid

end of turn score, returning (if any) meeples
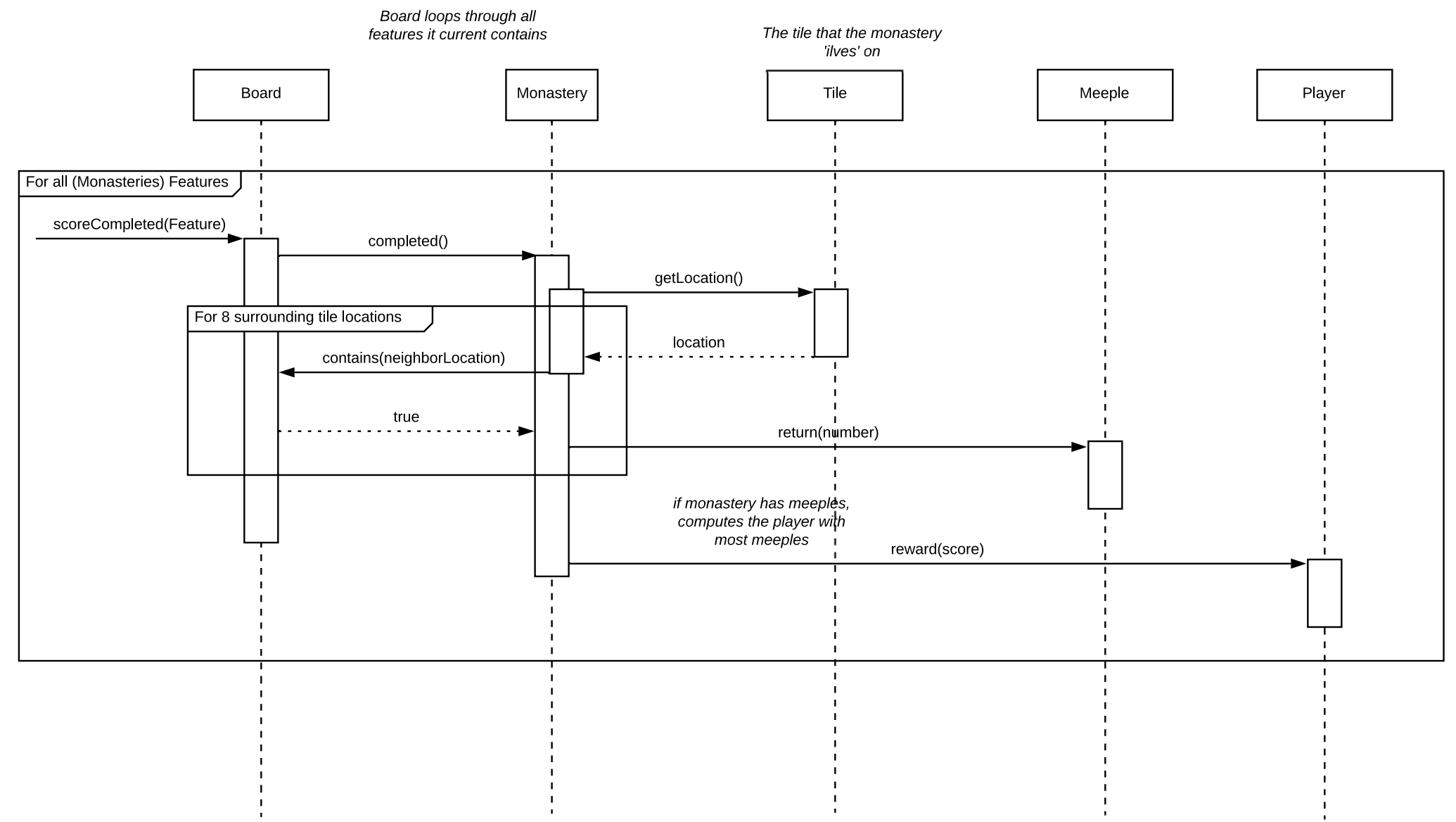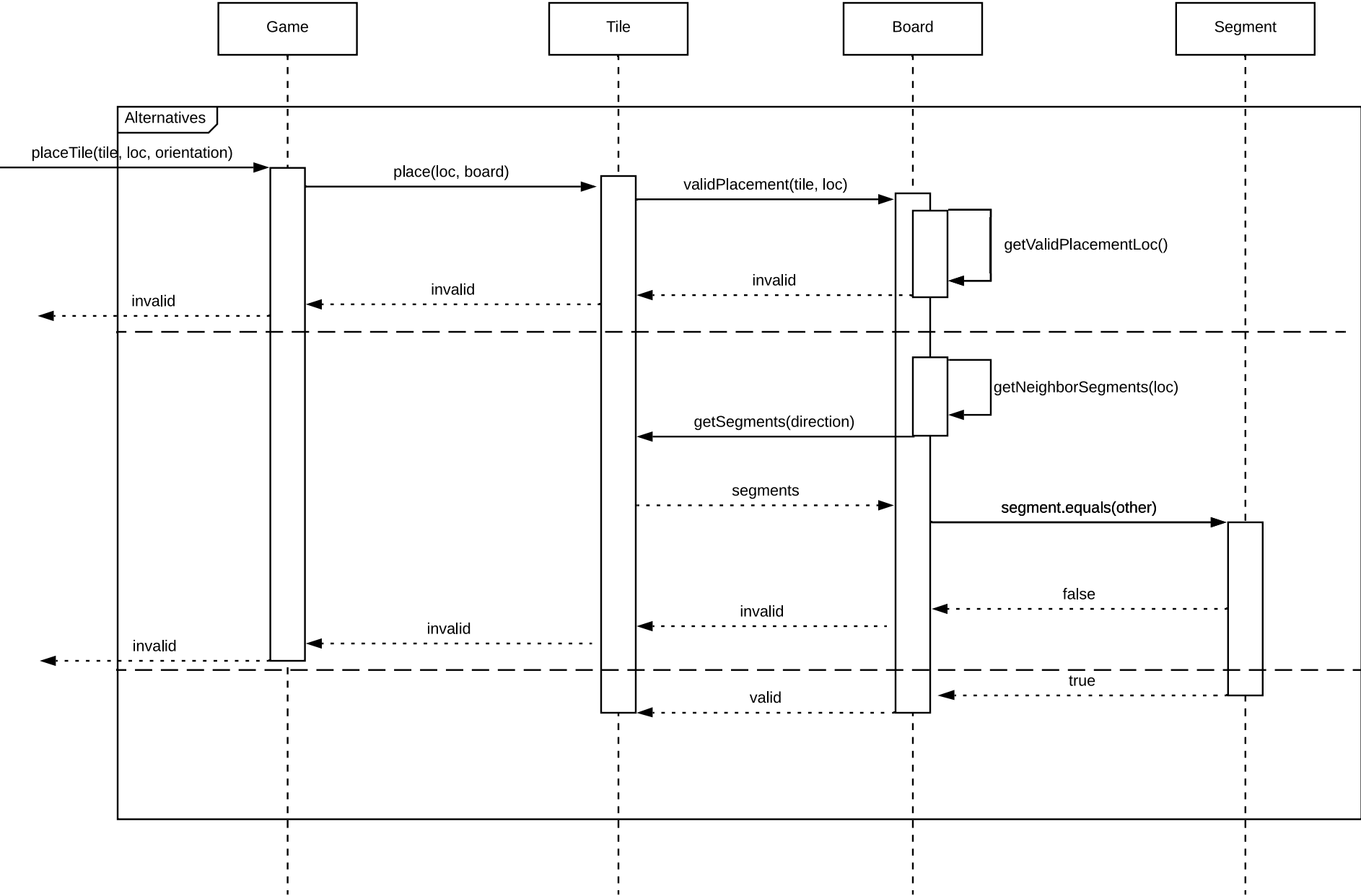
final score, winner

OPERATION: The user attempts to place a tile, without a meeple.

PRECONDITIONS: It is the player's turn to play. The location of the attemted placement is unoccupied by tiles, but has at least one neighboring tile. The segment(s) on the tile to be placed aligns with their corresponding neighboring segments.

POSTCONDITIONS: The new tile and the location of its placement was added to the game board. Additionally, had any segments on the new tiles completed one or more features, each completed feature's owner received a correspondig score which was added to their running score in the score board. The meeple(s) occupying any completed feature are no longer associated with the features and return to their corresponding players meeple supply.

INTERACTION: Game detects newly completed, previously played monasteries, determines whether they contain meeples, scores the monasteries as needed, and returns any scored meeples to their players.

Board loops through all features it current contains

The tile that the monastery 'ilves' on

Board	Monastery	Tile	Meeple	Player

For all (Monasteries) Features

scoreCompleted(Feature)

completed()

getLocation()

For 8 surrounding tile locations

contains(neighborLocation)

location

true

return(number)

if monastery has meeples, computes the player with most meeples

reward(score)

INTERACTION: Validation of tile
placement without a meeple

Game | Tile | Board | Segment

Alternatives

placeTile(tile, loc, orientation)

place(loc, board)

validPlacement(tile, loc)

getValidPlacementLoc()

invalid

invalid

invalid

getNeighborSegments(loc)

getSegments(direction)

segments

segment.equals(other)

false

invalid

invalid

invalid

true

valid

OBJECT MODEL
/////////////////////////////////

Board

- tiles: Map<Coord, Tiles>
- features: List<Feature>

+ validTilePlacement(tile: Tile)
+ actionAvailable(tile: Tile)
+ add(tile: Tile)
+ scoreCompletedFeature(feature: Feature)
+ scoreFinalFeatures(feature: Feature)
- mergeFeatures(features)

Game

- end: Boolean
- players: List<Players>
- curPlayer: Player
- board: Board
- currentTile: Tile
- turn: int

+ start()
+ hasEnded(): Boolean
+ startTurn()
+ endTurn()
+ getScore()
+ drawTile()
+ rotateTile()
+ placeTile()
+ placeMeeple()

**<<interface>>
Features**

+ score()
+ completed()
+ addMeeples()
+ validMeeplePlacement()

ExtendibleFeature

- numExtendibleSegments: int

+ abstract score()

Player

- name: String
- meeples: Meeple
- score: int

- getScore()

Tile

- location: Coord
- segments: List<Segment>
- features: List<Feature>
- orientation: List<Coord>

+ place(location: Coord)
+ rotate(dir: Direction)
+ getFeatures()
+ getLocation()
+ getSegment(dir: Segment)
+ getLinkedSegments(segments: List<Segments>)

Cities

- tiles: List<Tile>
- Map<Meeples, Integer>

+ score()
...

Roads

- tiles: List<Tile>
- Map<Meeples, Integer>

+ score()
...

Links to

Segment

- type: Enum
- loc: Array<Integer>
- featurePointer: Features

+ linkFeature(feature: Feature)
+ getLinkedFeatures()

Meeple

- master: Player
- feature: Feature
- placed: boolean

+ place(tile: Tile, segment: Segment)
+ return()

Monasteries

- tiles: List<Tile>
- Map<Meeples, Integer>

+ score()
...

1

1

1

1..*

1

0..*

0..*

1

1

1

1..*

5

1

0..7

0..*