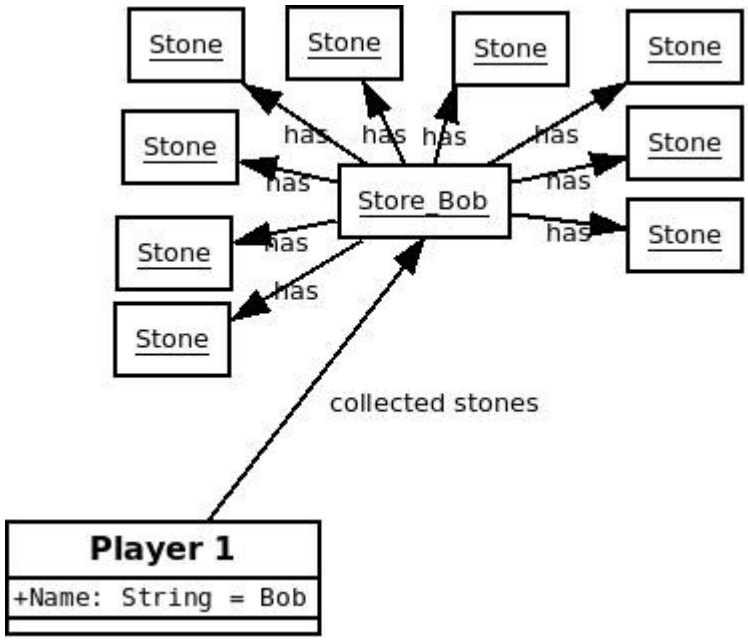
Mancala Test Cases

1) Returning number of captured stones

Precondition: game situation of player Bob is as follows:



Action: Instance method Player.getStoneNumber() is called out for player Bob.

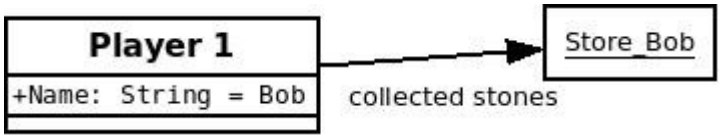
test = Bob.getStoneNumber();

Postcondition: Game situation does not change (exactly the same diagram as before). Number of captured stones is stored in the variable test.

test = 9;

2) Returning number of captured stones in case of empty mancala

Precondition: game situation of player Bob is as follows:

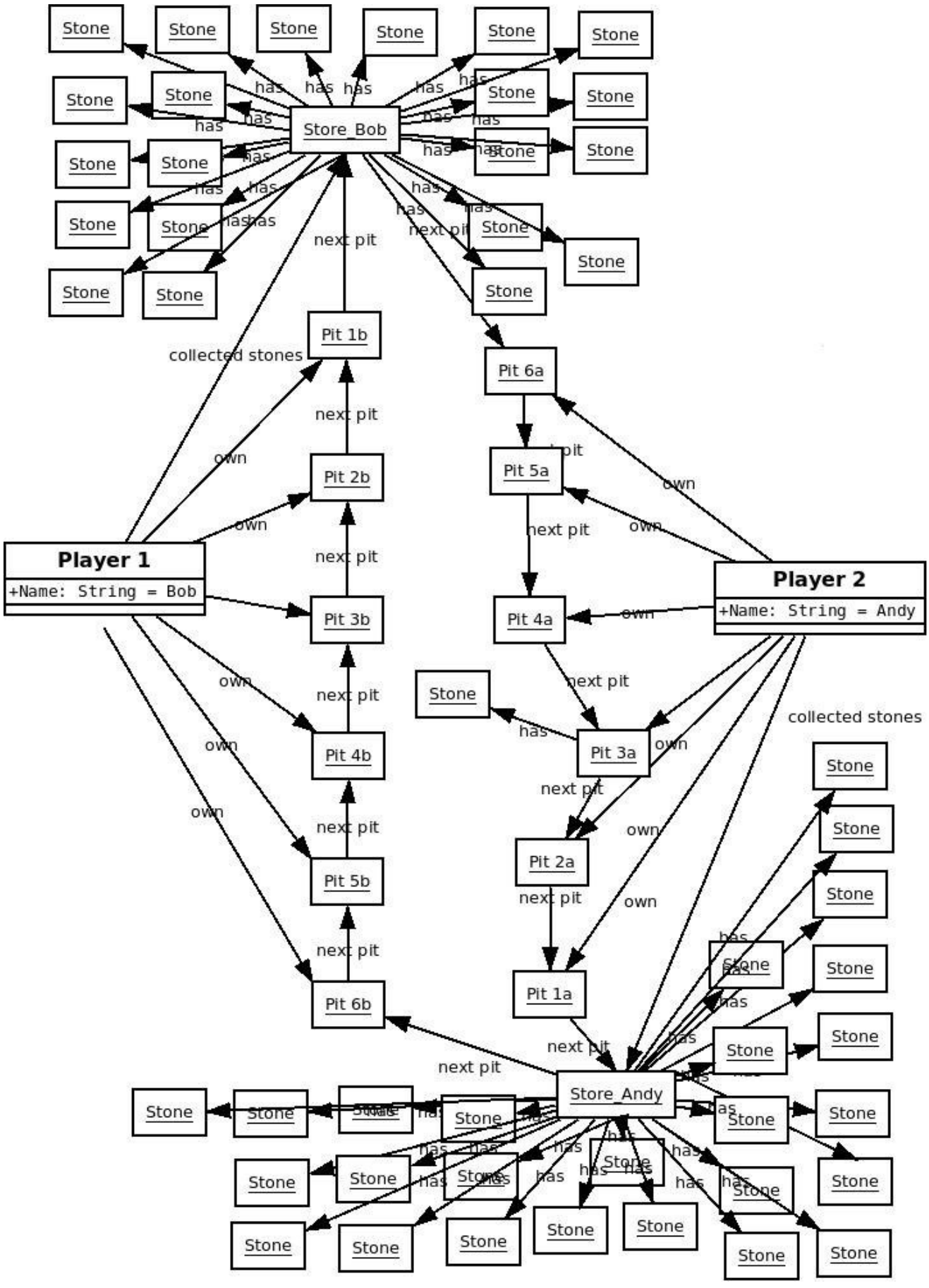


Action: Instance method Player.getStoneNumber() is called out for player Bob.
test = Bob.getStoneNumber();

Postcondition: Game situation does not change (exactly the same diagram as before). Number of captured stones is stored in the variable test. Test = 0;

3) Controlling whether the game is over

Precondition: game situation is as follows:

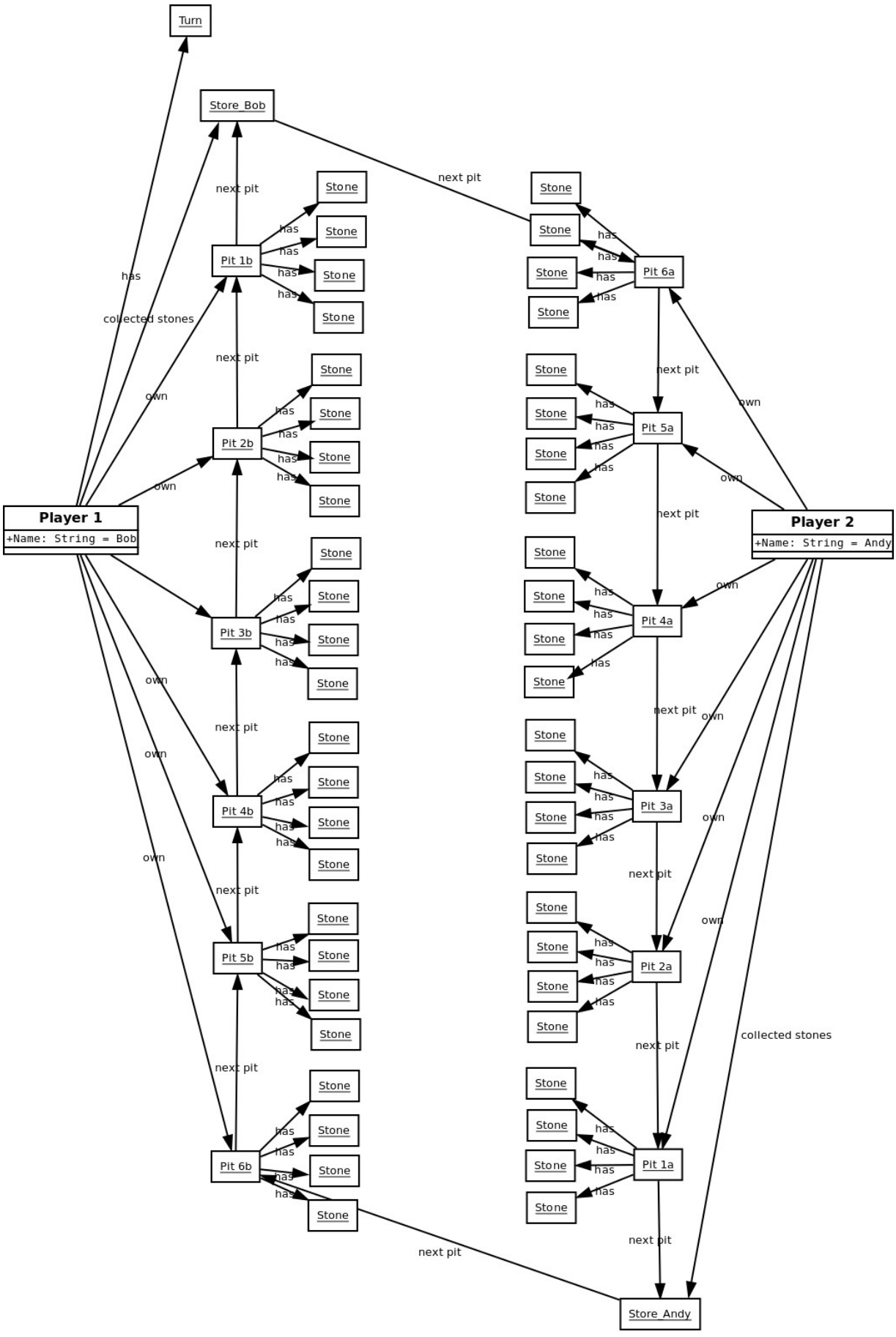

Player 1
+Name: String = Bob
Player 2
+Name: String = Andy
Store_Bob
Store_Andy
Pit 1b
Pit 2b
Pit 3b
Pit 4b
Pit 5b
Pit 6b
Pit 6a
Pit 5a
Pit 4a
Pit 3a
Pit 2a
Pit 1a
Stone
has
own
next pit
collected stones

Action: method Turn.isGameOver() is called out
check = Turn.isGameOver();

Postcondition: Game situation does not change (exactly the same diagram as before). check = True;

4) Controlling whether the game is over

Precondition: game situation is as follows:

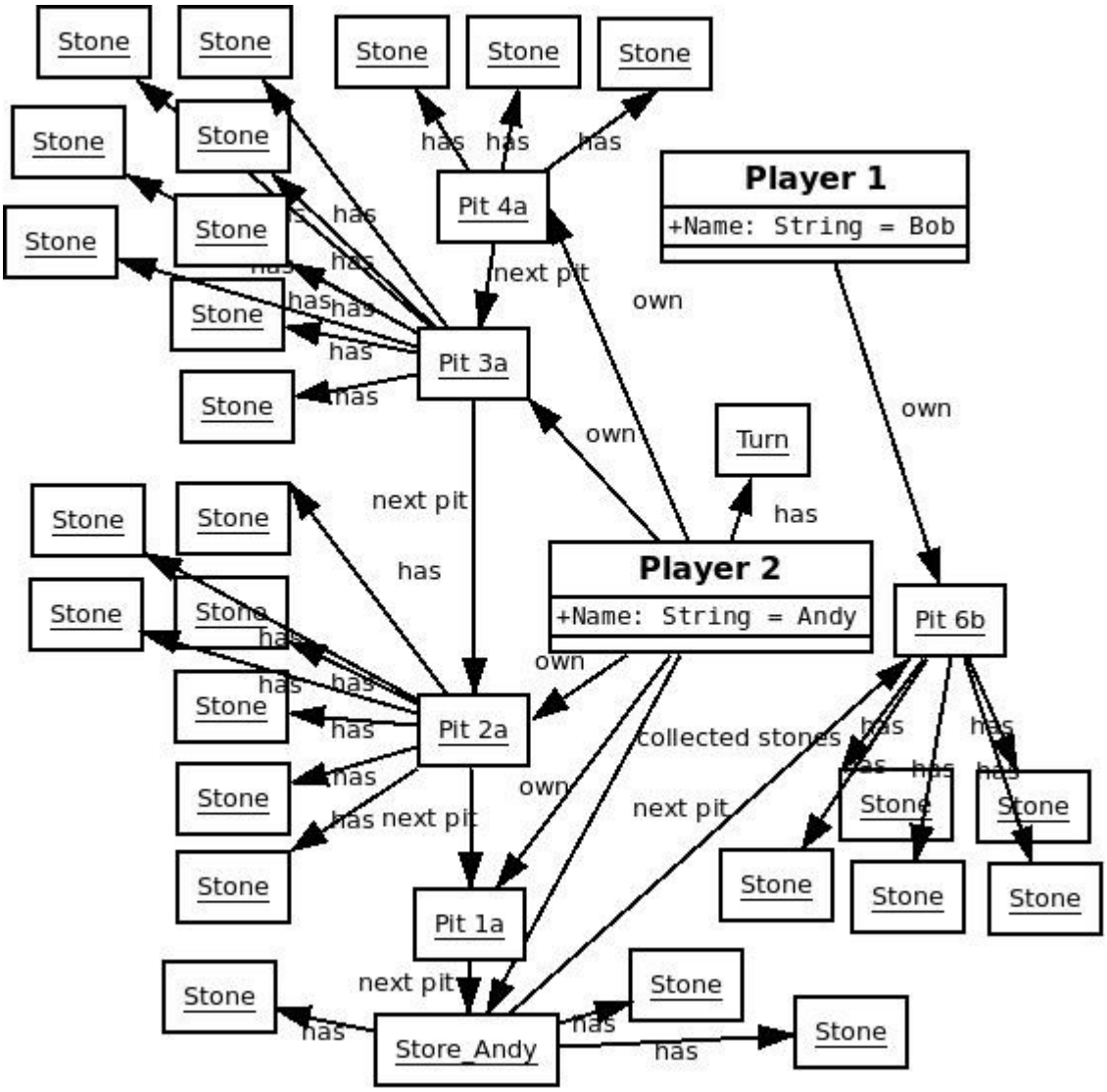

Turn
Store_Bob
next pit
Stone
Pit 1b
Pit 2b
Pit 3b
Pit 4b
Pit 5b
Pit 6b
has
collected stones
own
Player 1
+Name: String = Bob
Pit 6a
Pit 5a
Pit 4a
Pit 3a
Pit 2a
Pit 1a
Player 2
+Name: String = Andy
Store_Andy

Action: method Turn.isGameOver() is called
check = Turn.isGameOver();

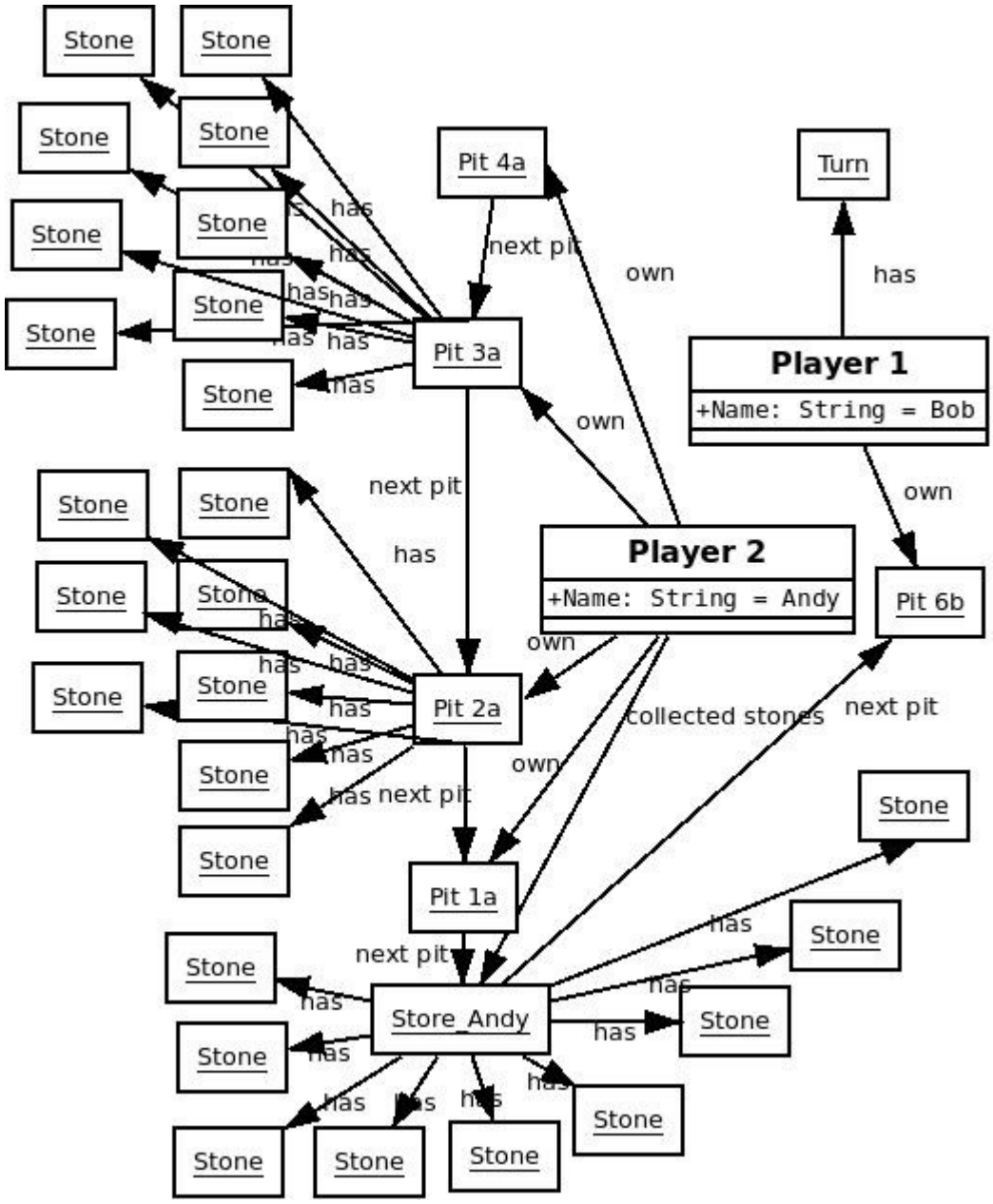
Postcondition: Game situation does not change (exactly the same diagram as before).
check = False;

5) Capturing opponent's stones

Precondition: game situation is as follows:

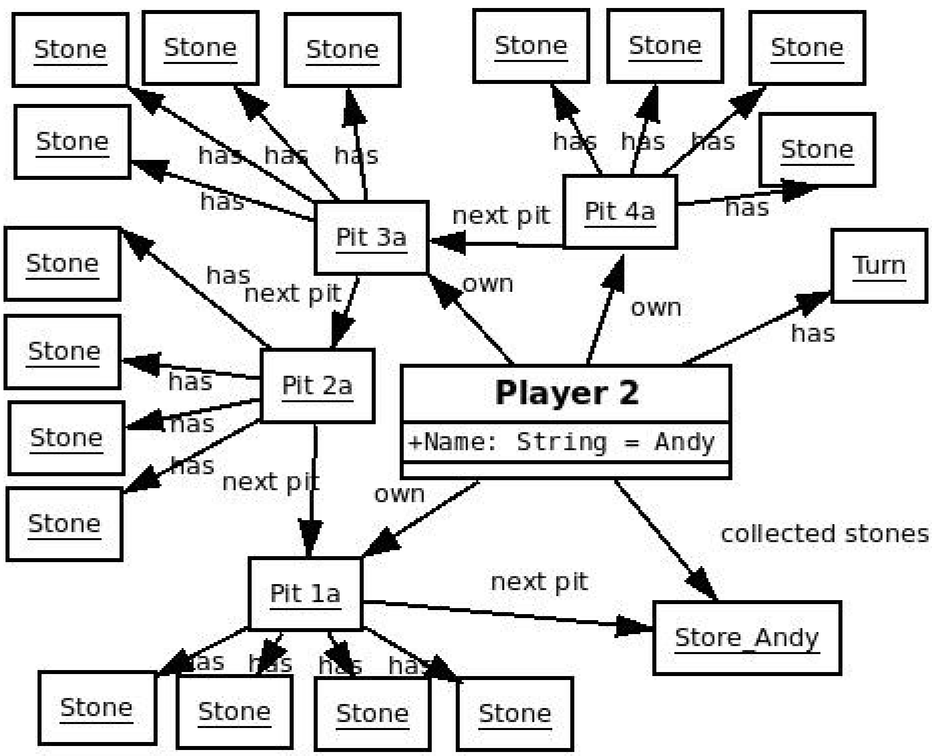


Action:
Player Andy plays pit 4. Andy.play(4);
Postcondition: game situataion is as follows. Any of the other objects didn't change.



6) Getting a free turn

Precondition: game situation of player Andy:

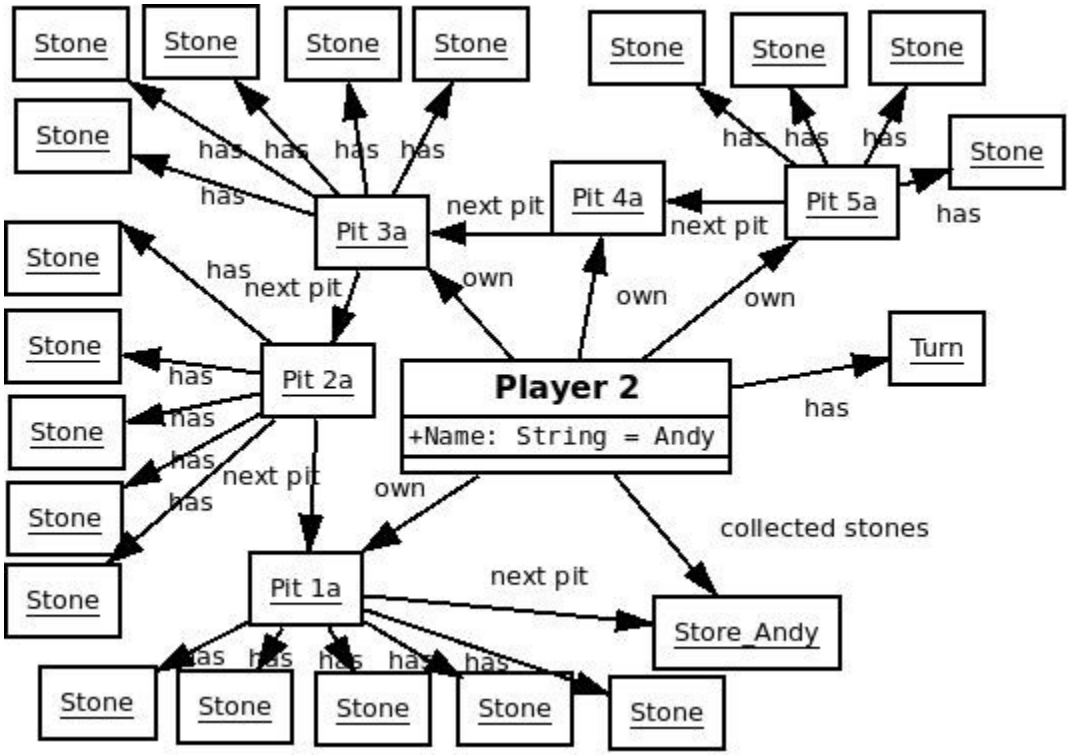


Action:
Player Andy plays pit 4. Andy.play(4);

Postcondition:
Player Andy still has turn.

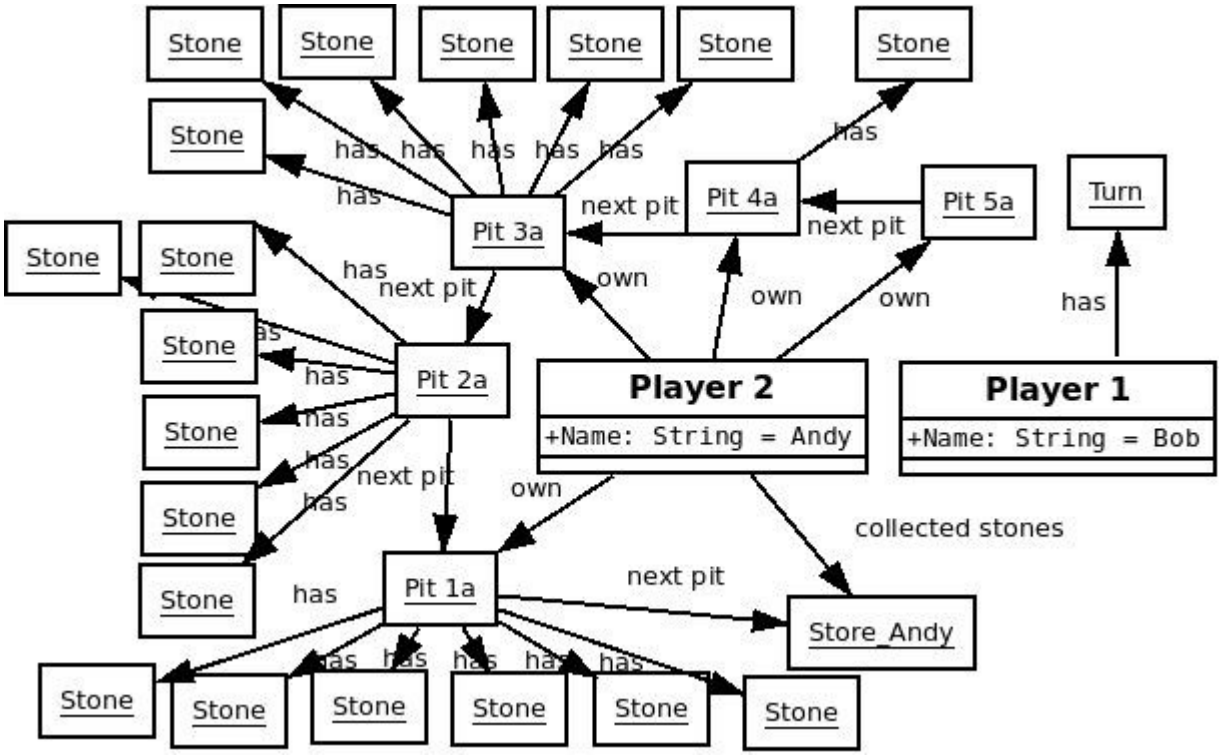
7) Player makes turn, last stone falls into his own pit

Precondition: game situation of player Andy:



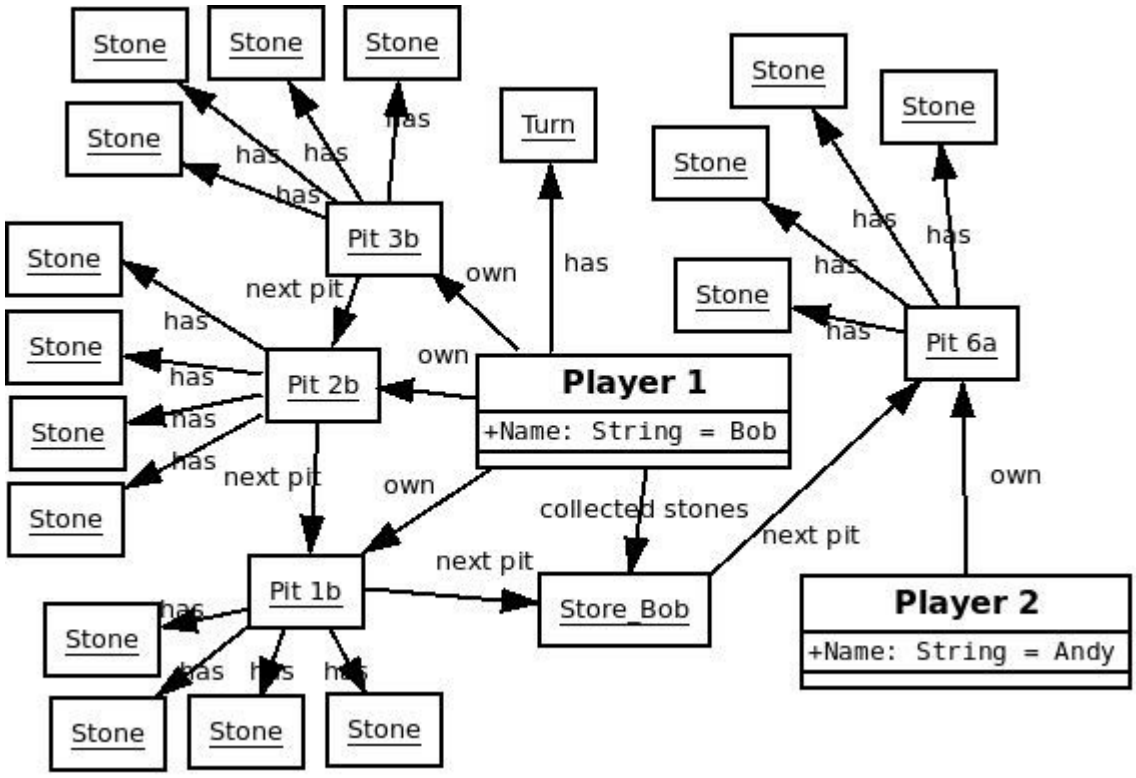
Action:
Player Andy plays pit 5. Andy.play(5);

Postcondition:
Game situation is as follows. Turn goes to player Bob.



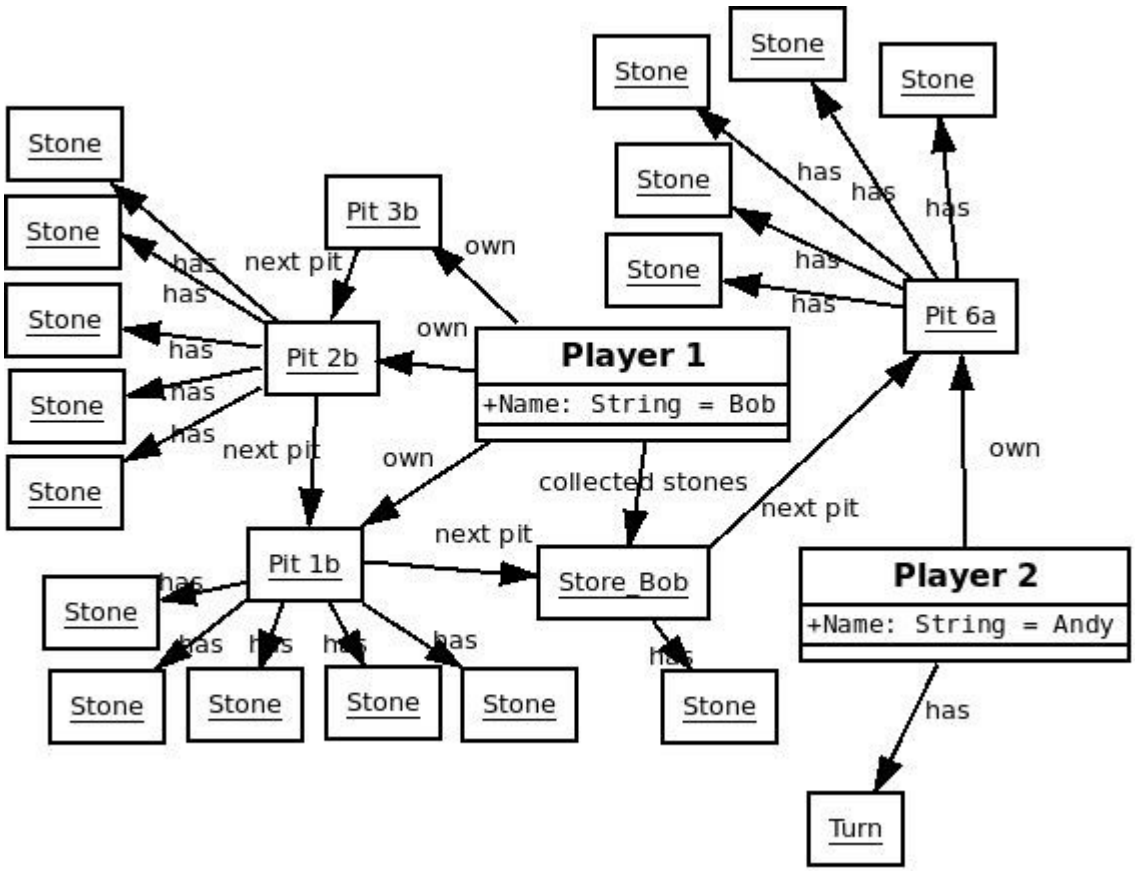
8) **Player makes turn, last stone falls into opponent's pit.**

Precondition: game situation is as follows:



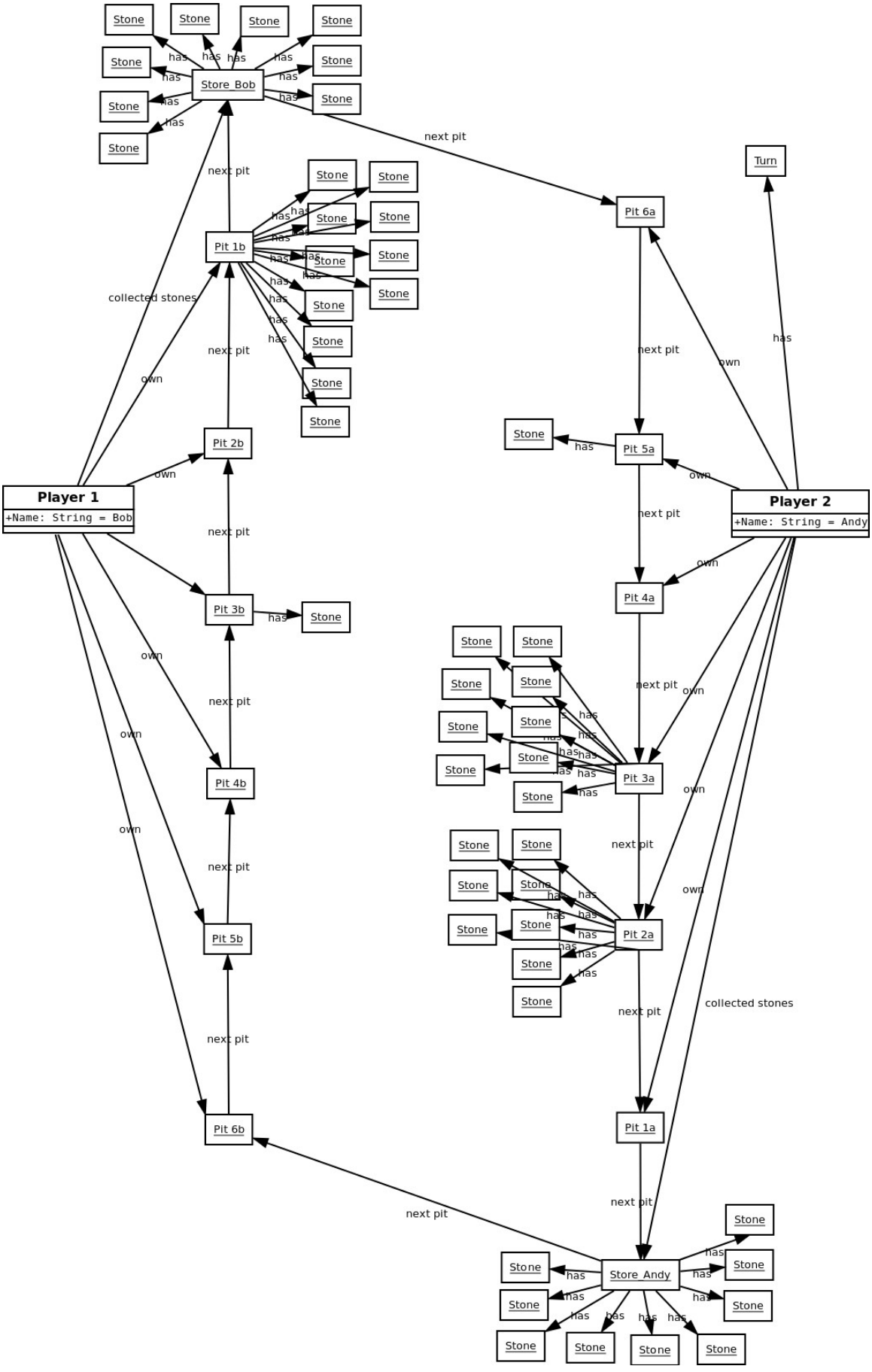
Action:
Player Bob plays pit 3. Bob.play(3);

Postcondition:
Game situation is as follows. Turn goes to player Andy.



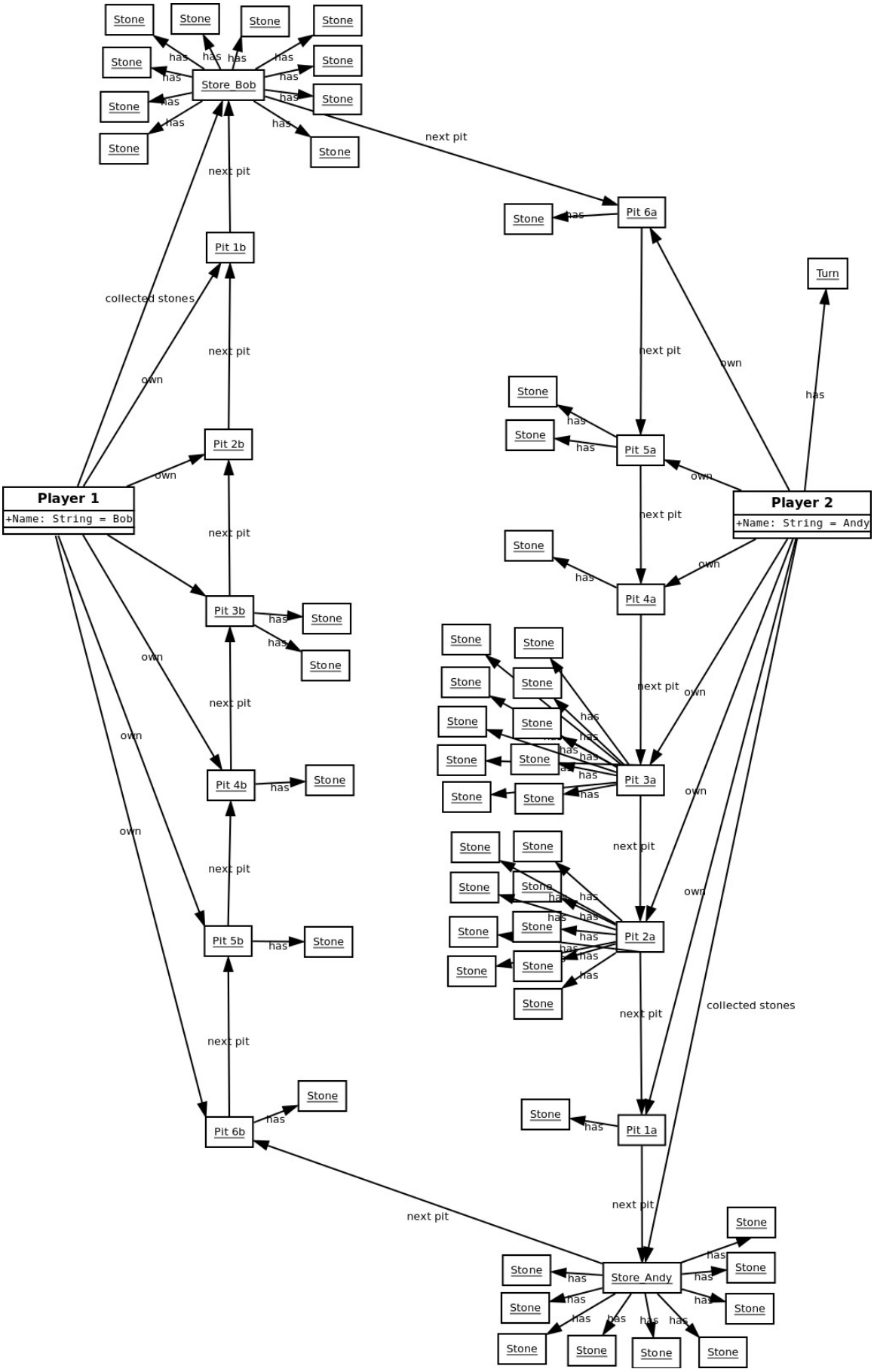
9) Player makes turn, stones cycle across the board and last stone falls into the initial player's pit.

Precondition: game situation is as follows:


Player 1
+Name: String = Bob
Player 2
+Name: String = Andy
Store_Bob
Store_Andy
Turn
Pit 1b
Pit 2b
Pit 3b
Pit 4b
Pit 5b
Pit 6b
Pit 6a
Pit 5a
Pit 4a
Pit 3a
Pit 2a
Pit 1a
Stone
has
own
next pit
collected stones

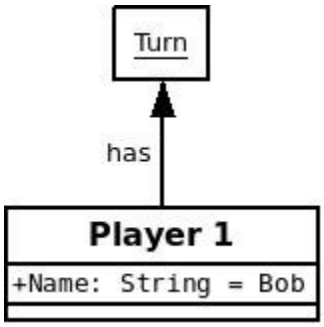
Action:
Player Bob plays pit 1. Bob.play(1);

Postcondition:
Game situation is as follows. Turn goes to player Andy.


Player 1
+Name: String = Bob
Player 2
+Name: String = Andy
Turn
Store_Bob
Store_Andy
Pit 1b
Pit 2b
Pit 3b
Pit 4b
Pit 5b
Pit 6b
Pit 1a
Pit 2a
Pit 3a
Pit 4a
Pit 5a
Pit 6a
Stone
has
next pit
own
collected stones

10) Controlling whether Bob has turn

Precondition: game situation of player Bob is as follows:

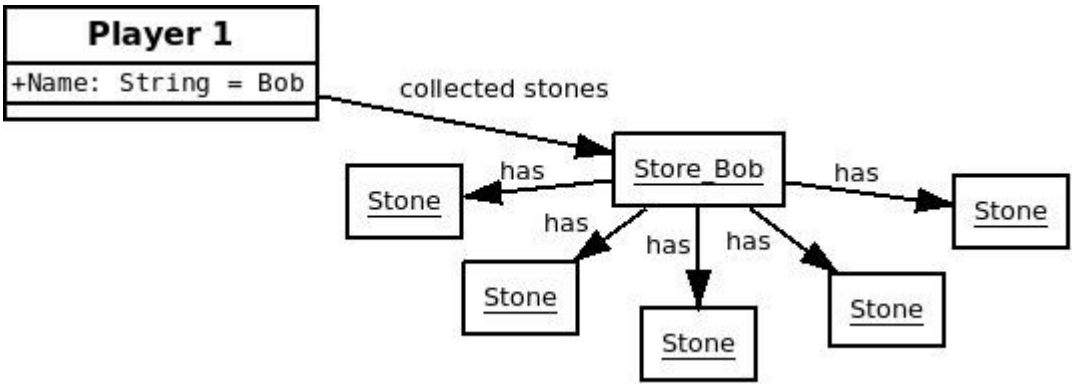


Action: method Bob.hasTurn() is called
check = Bob.hasTurn();

Postcondition: Game situation does not change (exactly the same diagram as before).
check = True;

11) Counting collected stones in a storage

Precondition: game situation of player Bob is as follows:

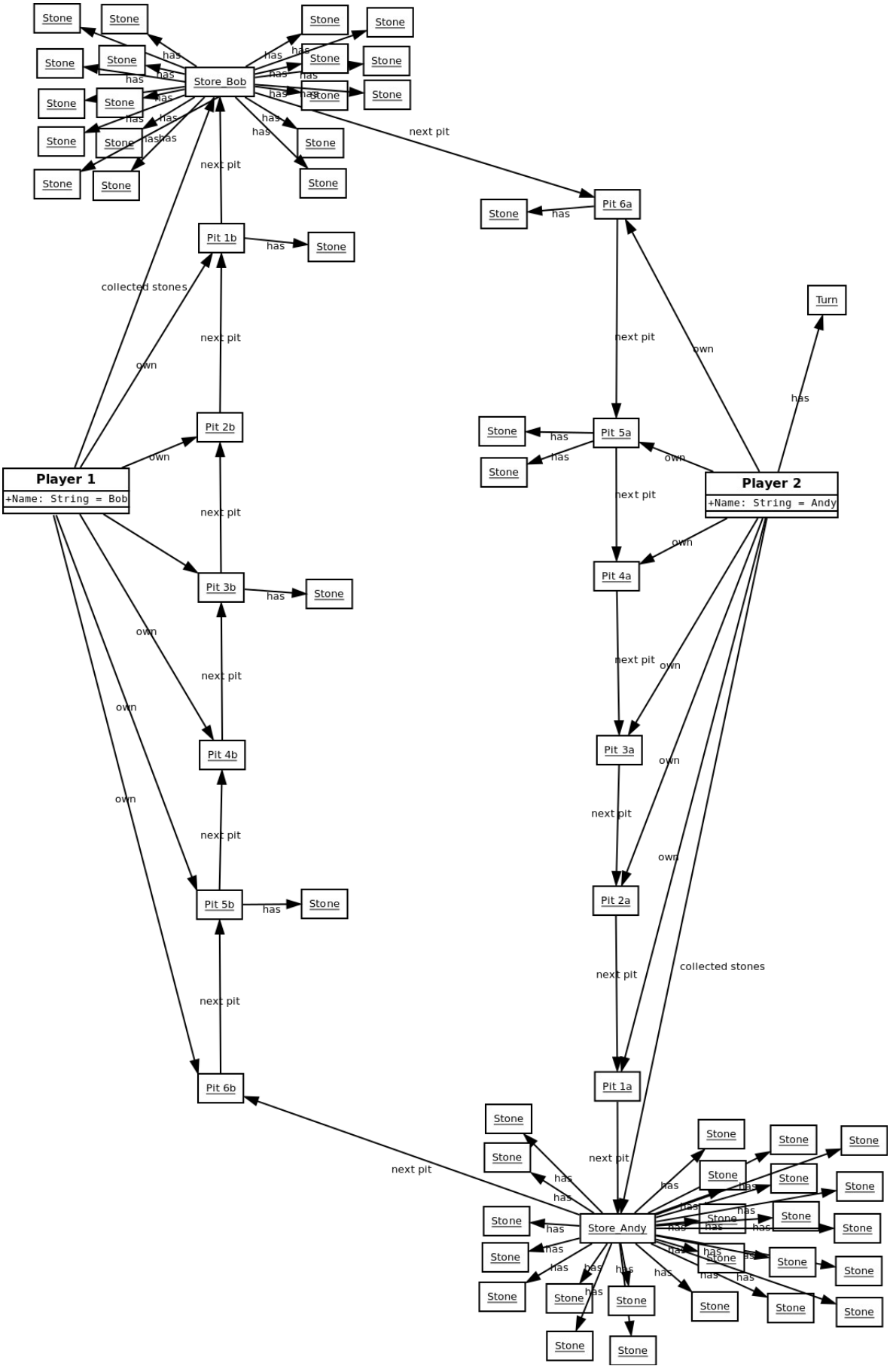


Action: method Bob.getPoints() is called
points = Bob.getPoints();

Postcondition: Game situation does not change (exactly the same diagram as before).
points = 5

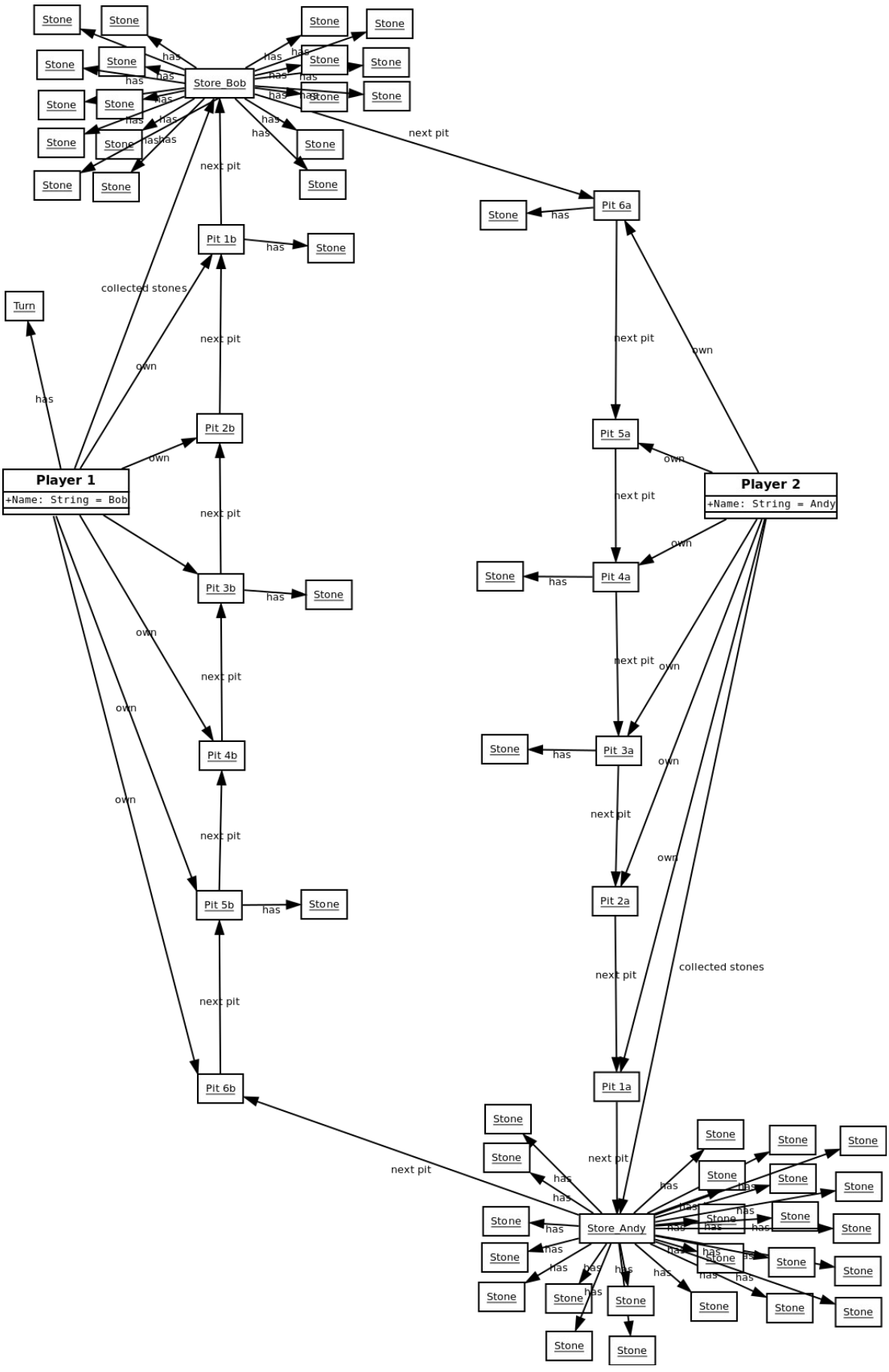
12) Stone falls into an empty pit and opponent's opposite pit is empty

Precondition: game situation is as follows:


Player 1
+Name: String = Bob
Player 2
+Name: String = Andy
Turn
Store_Bob
Store_Andy
Pit 1b
Pit 2b
Pit 3b
Pit 4b
Pit 5b
Pit 6b
Pit 6a
Pit 5a
Pit 4a
Pit 3a
Pit 2a
Pit 1a
Stone
has
own
next pit
collected stones

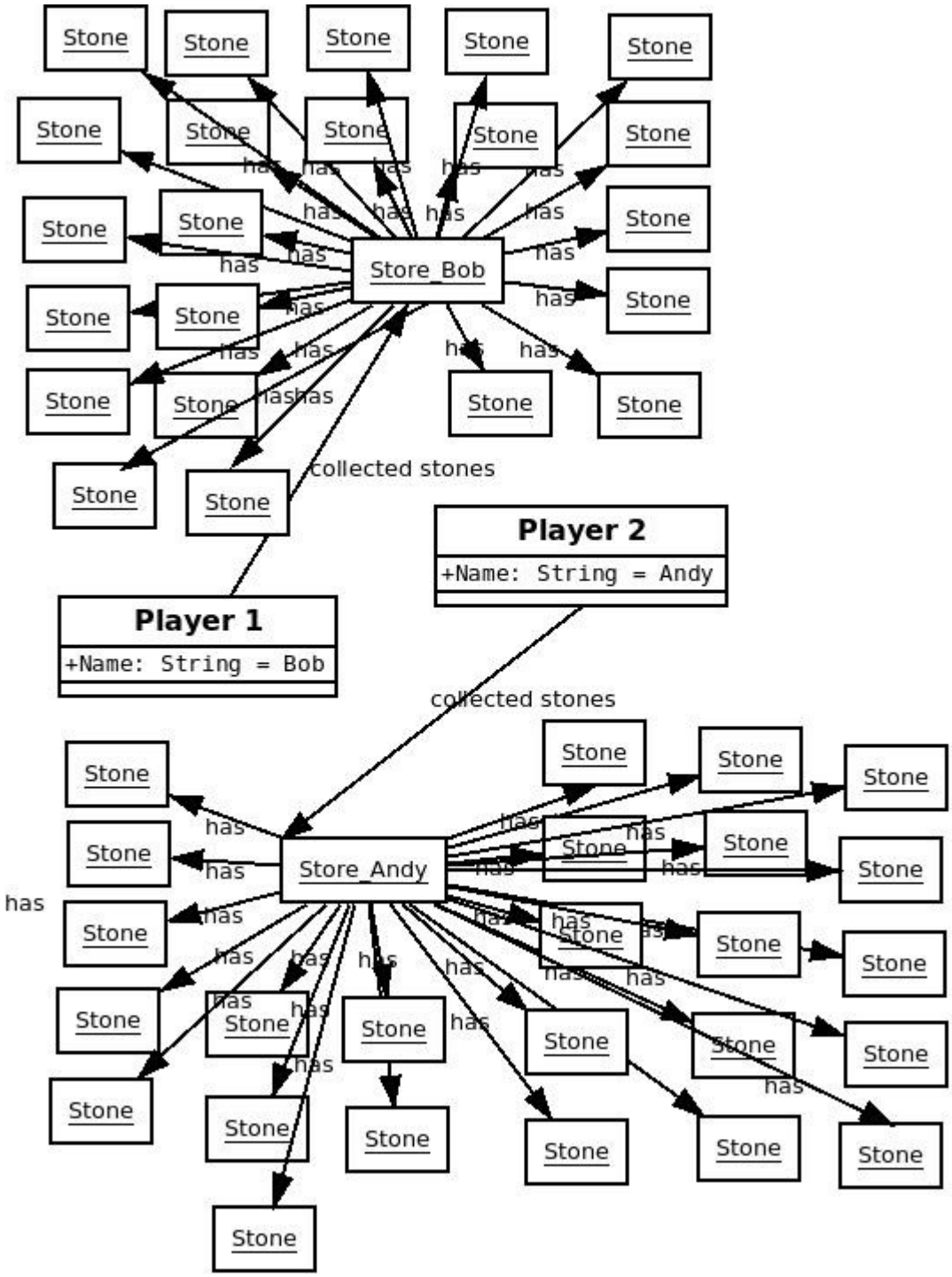
Action:
Player Andy plays pit 5. Andy.play(5);

Postcondition:
Game situation is as follows. No stones are captures. Turn goes to player Bob.


Player 1
+Name: String = Bob
Player 2
+Name: String = Andy
Turn
Store_Bob
Store_Andy
Pit 1b
Pit 2b
Pit 3b
Pit 4b
Pit 5b
Pit 6b
Pit 6a
Pit 5a
Pit 4a
Pit 3a
Pit 2a
Pit 1a
Stone
has
next pit
own
collected stones

13) Checking who is the winner of the game

Precondition: game situation is as follows:

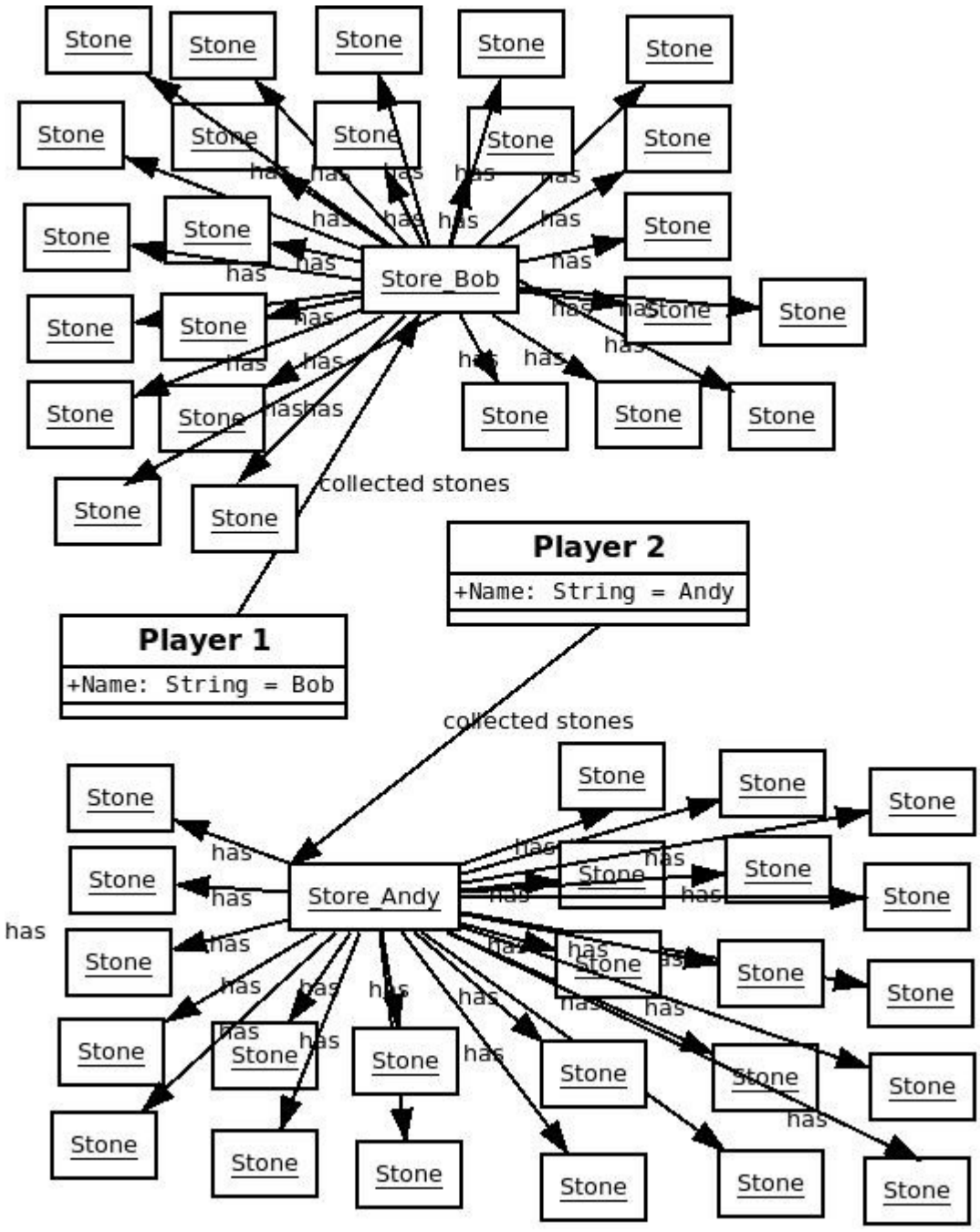


Action: method getWinner(Player, Player) is called
winner = getWinner(Bob, Andy);

Postcondition: Game situation does not change (exactly the same diagram as before).
winner = 2 (which means that Andy is the winner)

14) Checking who is the winner of the game

Precondition: game situation is as follows:



Action: method getWinner(Player, Player) is called
winner = getWinner(Bob, Andy);

Postcondition: Game situation does not change (exactly the same diagram as before).
winner = 0 (which means game ended in a draw)