SWEN30006
Software Modelling and Design

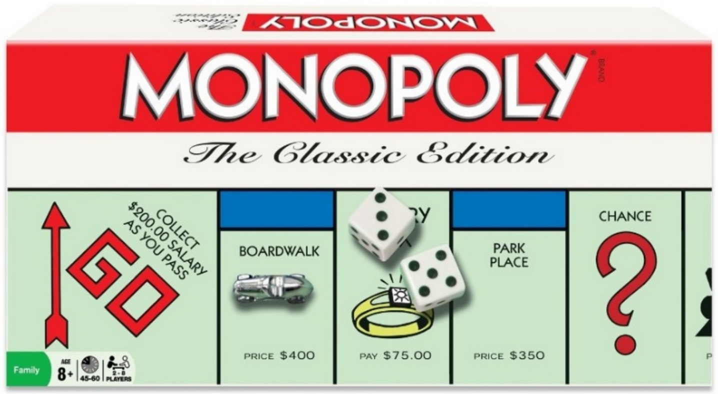
MONOPOLY ITERATION 3

Larman Chapters 27, 31 and 36

I think it's wrong that only one company makes the game Monopoly.

—Steve Wright

Case 2: Monopoly Game System



A Software Simulation of Monopoly

User starts off game and watches the activities of the simulated players.

Requirements in Iteration-3

- New square types: Lots, Railroads, and Utilities.
- When a player P lands on one of these:
  - If it is not owned, P may buy it.
    - the price of the square is deducted from P's money
    - P becomes the owner of the square
  - If it is owned by P, nothing happens.
  - If it is owned by a player other than P, P must pay the owner rent.

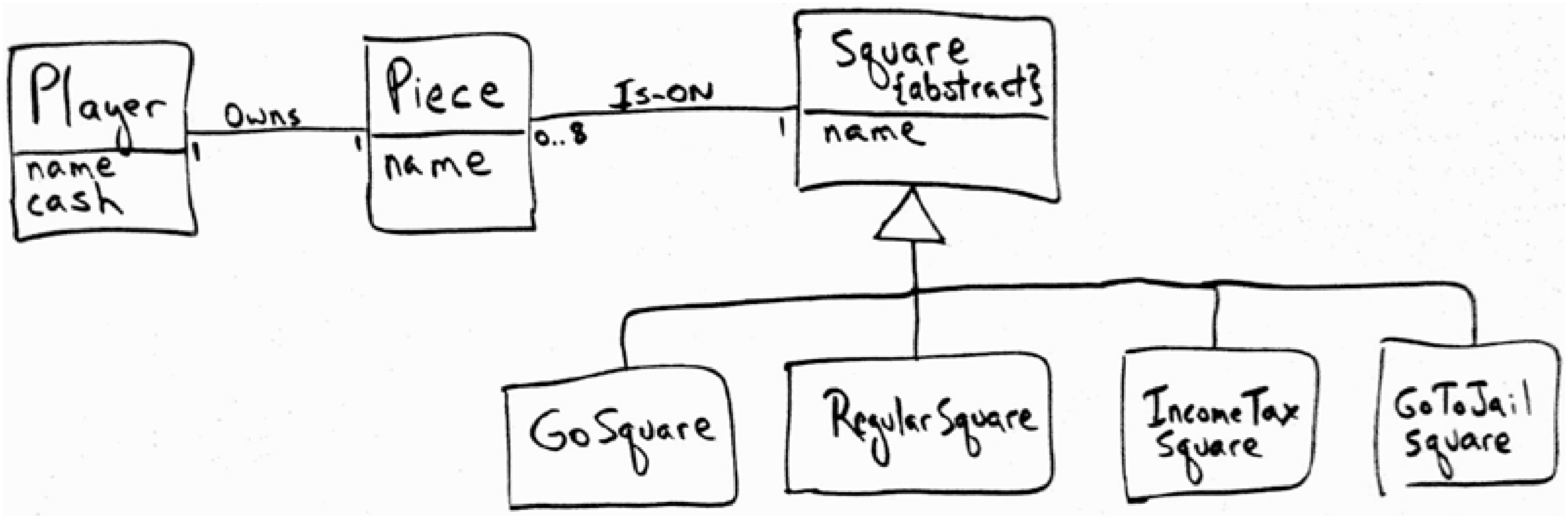
Rent

- The rent calculations are:
  - Lot rent is (board position) dollars; e.g., if position 5, then $5.
  - Railroad rent is 25 dollars times the number of Railroads owned by the owner
    - e.g., if own 3 Railroads, then $75.
  - Utilities rent is 4 times the number shown on the dice when the player lands on the square (do not roll again)

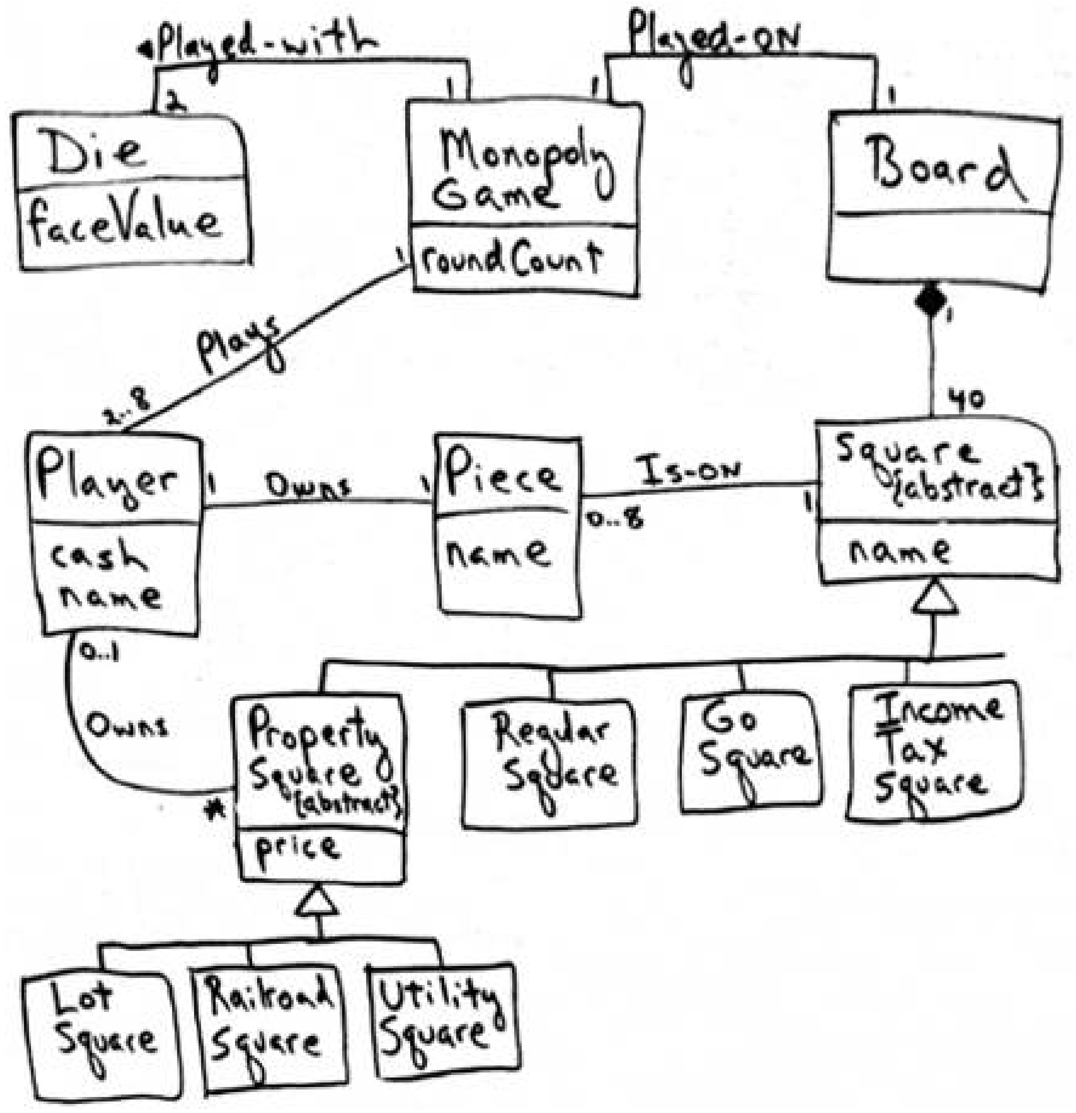
Solution Output Example

```
$ java -jar monopoly_it3.jar
        com.unimelb.swen30006.monopoly.MonopolyGame.main
Please enter the number of players (between 2 - 8): 2
Player 1: dice total = 5 move to Railroad
Player 1 buy Railroad
Player 2: dice total = 7 move to Square 7
Player 1: dice total = 7 move to Utility
Player 1 buy Utility
Player 2: dice total = 5 move to Utility
Player 2 pays $20.0 to Player 1
Player 1: dice total = 8 move to Square 20
```

It2 Domain Model Changes



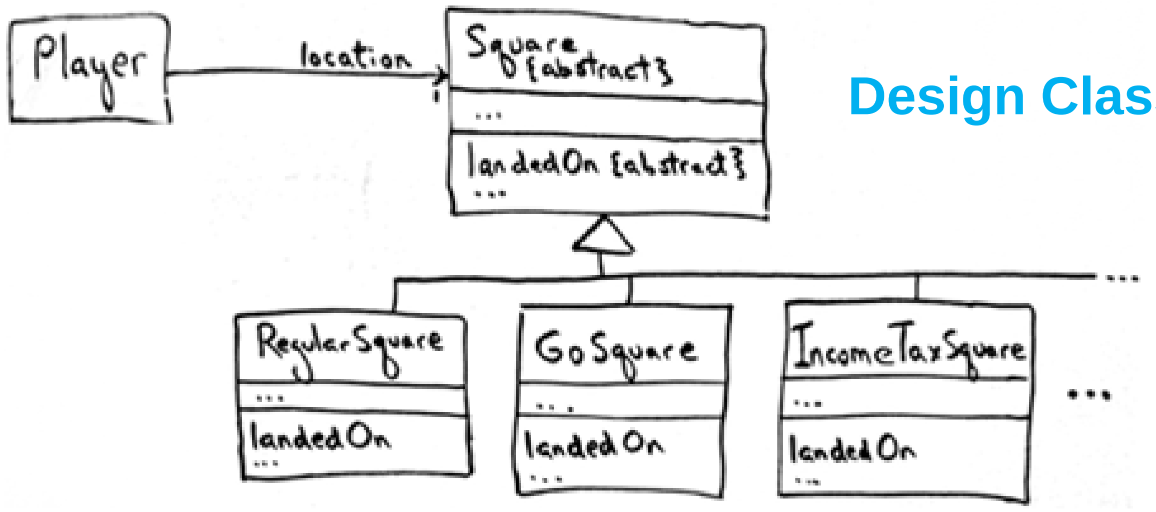
It3 Domain Model



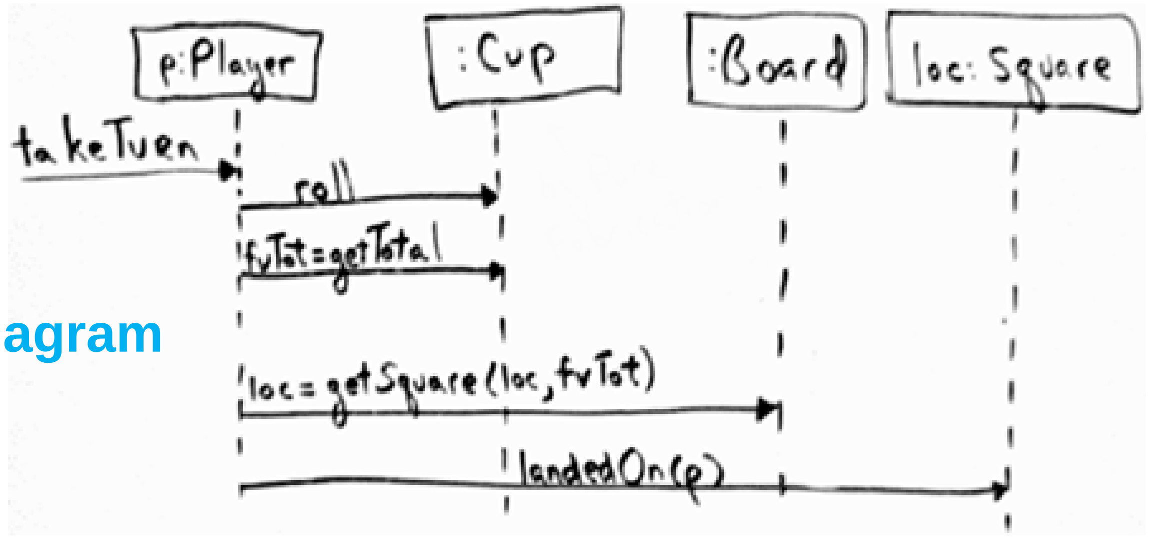
Essential Design Elements

- Polymorphism is applied
  - for each kind of square that has a different landed-on behavior, there is a polymorphic *landedOn* method
  - When a *Player* software object lands on a *Square*, it sends it a *landedOn* message

Polymorphic landedOn design strategy

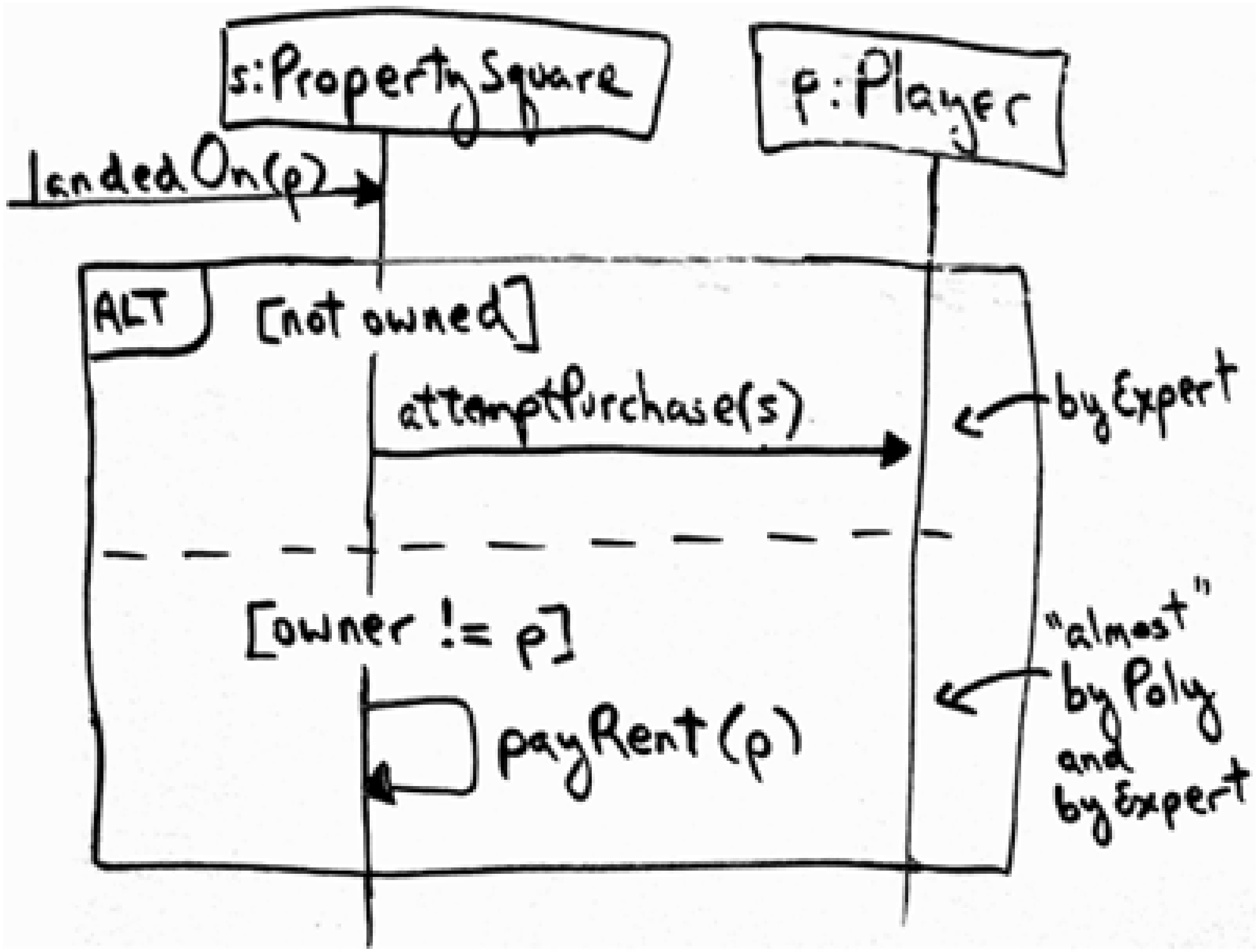


Design Class Diagram

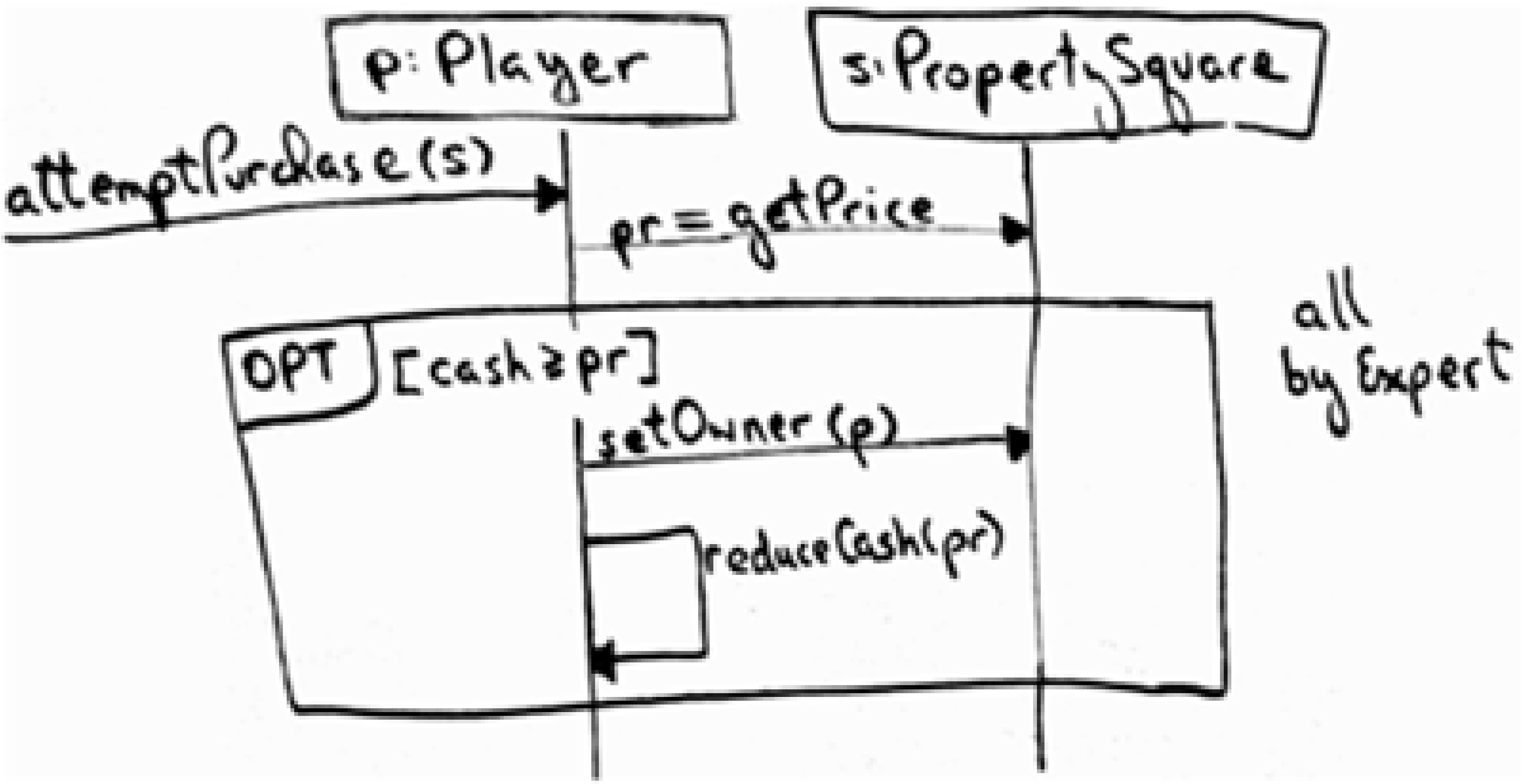


Sequence Diagram

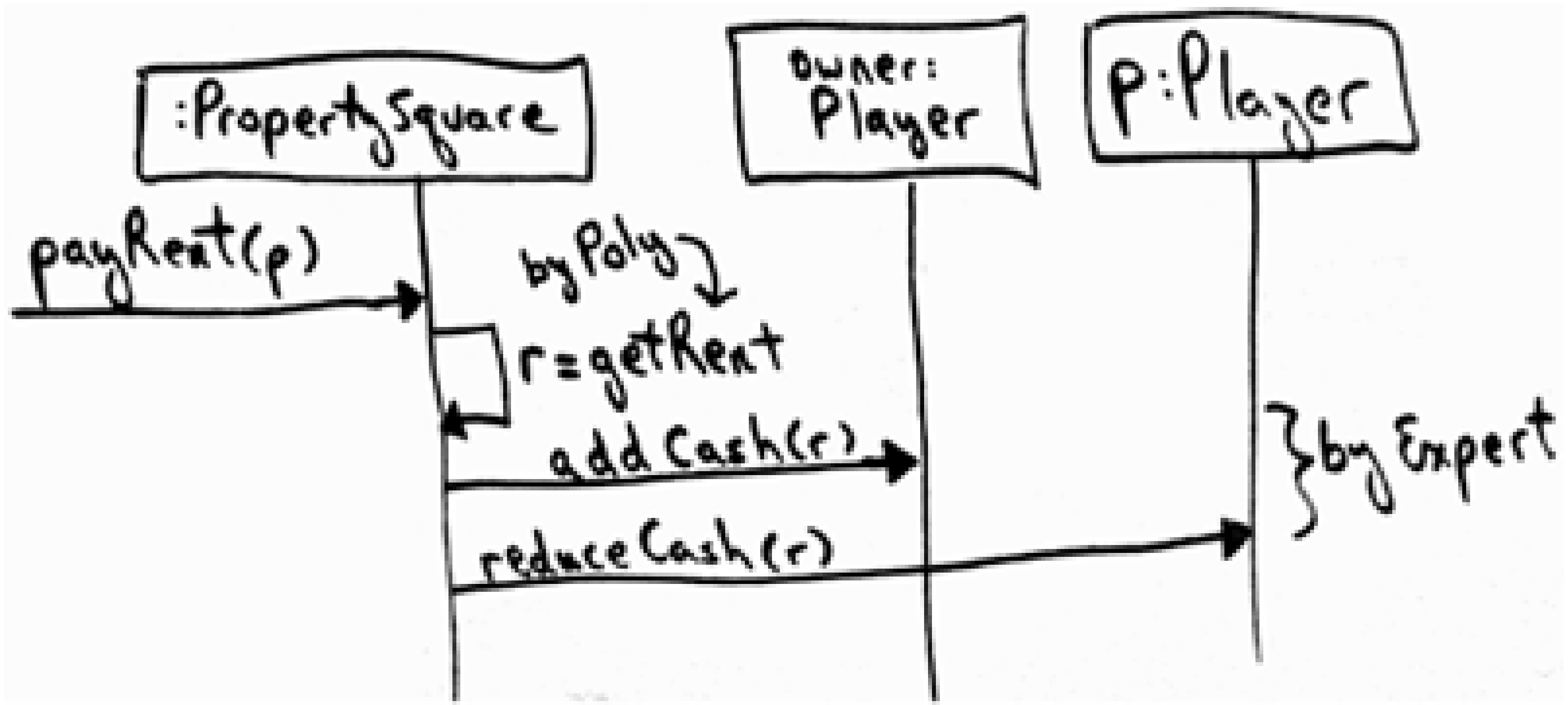
Landing on a PropertySquare



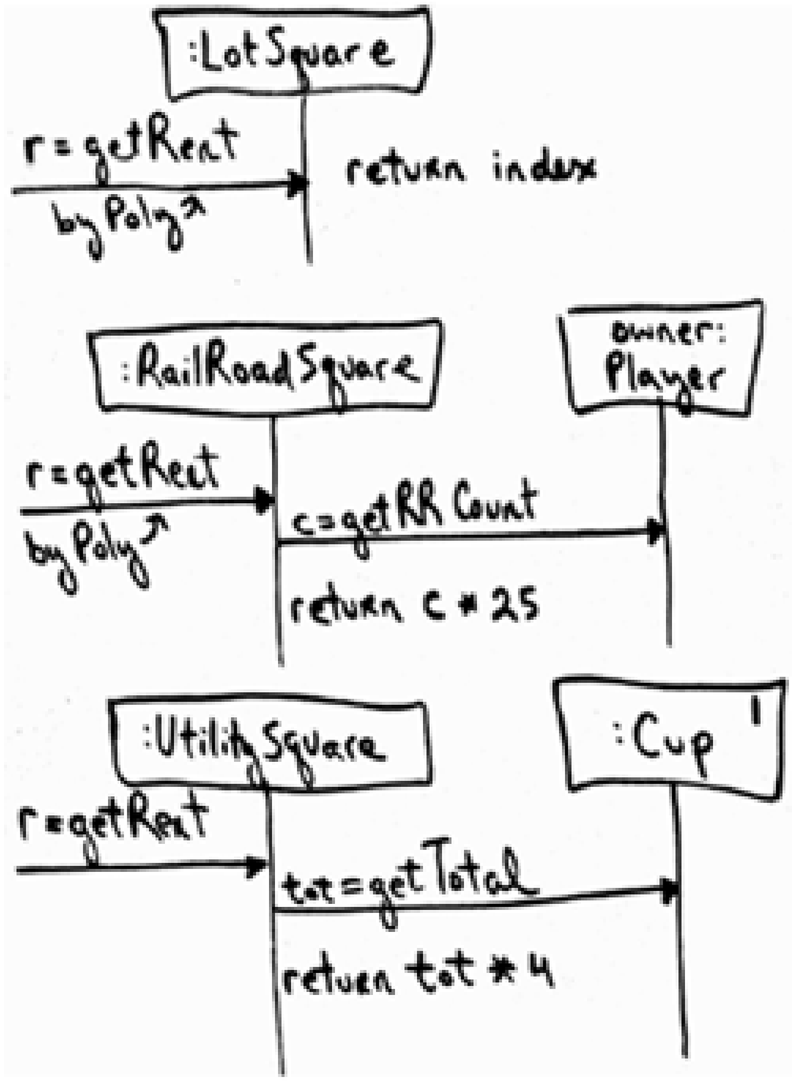
Attempting to purchase a property



Paying rent


:PropertySquare
owner: Player
p:Player
payRent(p)
by Poly
r=getRent
addCash(r)
reduceCash(r)
} by Expert

Polymorphic *getRent* methods


:LotSquare
r=getRent
byPolyα
return index
:RailRoadSquare
owner: Player
r=getRent
byPoly↗
c=getRRCount
return c * 25
:UtilitySquare
:Cup
r=getRent
tot=getTotal
return tot * 4