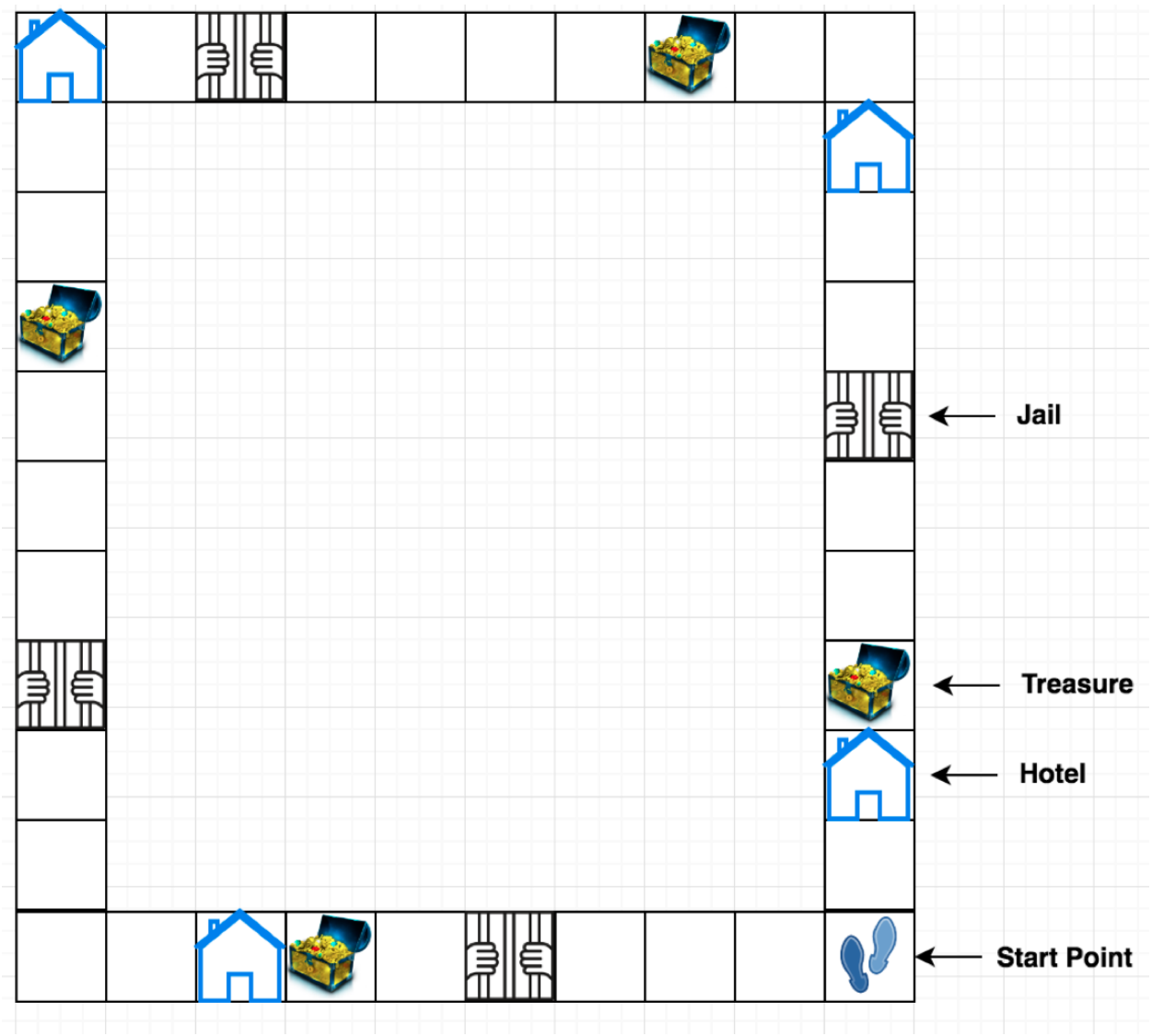
Business House Game

Business house is a board game which require minimum two players. Player uses a random number between 2-12 and move on the board accordingly.
Each player has some amount at start.

Below image represent a game board.



The board cell may be one of the following types.

1. Jail : When user lands on it, a defined amount, for e.g. Rs 150, will be deducted from user's money.
2. Treasure : When user lands on it, a defined amount, for e.g. Rs 200, will be added to user's money.
3. Hotel : This is a special type of entity.
    1. A hotel is of a defined worth, for e.g: Rs. 200.
    2. When user lands on it and has required money, he has to buy it.
    3. If any other user lands on a pre owned hotel, user needs to pay Rs.50 to hotel owner.

How To Play :

1. Two+ users will start from starting point with initial money.
2. They have to move as per random number between 2-12.
3. Every move has to follow cell type rules defined above.
4. Maximum ten chances will be awarded to each player.
5. After ten chances, player with maximum money, will be declared as winner.

Inputs :

Number of Players : 3

Cells Position and Type :

E,E,J,H,E,T,J,T,E,E,H,J,T,H,E,E,J,H,E,T,J,T,E,E,H,J,T,H,J,E,E,J,H,E,T,J,T,E,E,H,J,T,E,H,E

E → EMPTY
J → JAIL
T → TREASURE H → HOTEL

Dice Output:

4,4,4,6,7,8,5,11,10,12,2,3,5,6,7,8,5,11,10,12,2,3,5,6,7,8,5,11,10,12

Initial Money for each player : 1000
Hotel Worth : 200
Hotel Rent : 50
Jail Fine: 150
Treasure Value : 200

Desired Output :

We need to print total worth of all players in decreasing order. Total worth means: money remaining at end + value of all hotel bought by player.

Player-1 has total worth 1200
Player-2 has total worth 1200
Player-3 has total worth 1050

Note:

- For the solution, we request that you use Java, Ruby, C#, .Net, Python, Clojure, Scala or JavaScript.
- There must be a way to supply the application with the input data via text file or console
- The application must run
- You should provide sufficient evidence that your solution is complete by indicating that it works correctly against the supplied test data

Rules:

- Having good unit test coverage is desirable. Also concepts from DDD (domain driven design) can be applied where ever suitable.
- You may not use any external libraries to solve this problem, but you may use external libraries or tools for building or testing purposes. Specifically, you may use unit-testing libraries or build tools available for your chosen language (e.g., JUnit, Ant, NUnit, Rspec, Rake, etc.).
- **In short – your code should be Well Designed, Object Oriented, Extendable & must have followed Code Quality standards.**