ConsilX is a disruptive technology start up developing scalable solutions for the life sciences and health care domain.

At ConsilX, we strive to build high quality software that has strong aesthetics (is readable and maintainable), has extensive safety nets to safeguard quality, handles errors gracefully, handle scale and works as expected, without breaking down, with varying input.

We are looking for people who can write code that has extensibility and maintainability built in, by adhering to the principles of Object Oriented Development, and have the ability to deal with the real-life constraints / trade-offs while designing a system.

It is important to note that we are not looking for a GUI and we are not assessing you on the capabilities around code required to do the I/O. The focus is on the overall design. So, while building a solution, it would be nicer if input to the code is provided either via unit tests or a file. Using command line (for input) can be tedious and difficult to test, so it is best avoided.

Following is a list of things to keep in mind, before you submit code, to ensure that your code focuses on attributes, we are looking for -

- Is behavior of an object distinguished from its state and is the state encapsulated?
- Have you applied SOLID principles to your code?
- Have you applied principles of YAGNI and KISS?
- Have you unit tested your code or did TDD? If you have not, we recommend you read about it and attempt it with your solution. While we do not penalize for lack of tests, it is a definite plus if you write them.
- Have you looked at basic refactoring to improve design of your code?
- Finally, and foremost, are the principles applied in a pragmatic way. Simplicity is the strongest of the trait of a piece of code. However, easily written code may not necessarily be simple code.

PROBLEM STATEMENT

Bankerless Monopoly:

Monopoly is a classic board game where players roll two six-sided dice to move around the game board, buying and trading properties.

As the players moves around the board and land on properties, they can buy unowned properties, build houses, pay rent if the property belongs to another player, or mortgage properties to pay rent.

If a player owes more money than his/her assets can afford, he/she is declared bankrupt and is out of the game.

The game has a Banker who is trusted and is in charge of all the financial transactions involved when a player buys a property, mortgages a property, collects rents from opponents, or pays rent to opponents, etc. In this problem, we look to automate the banker's responsibilities on every player turn.

Each player is seeded with $2,000 at the beginning of the game by the banker. As the player crosses GO, he is seeded with $200.

For the sake of simplicity let's assume that the board has only eight property tiles, one special tile and three players are playing it. Every property has the cost value, color and its rent levels (1-5). Rent levels, start at 1, go up by 1 right after a successful rent payment for a property. Special tile - increases the rent level by 1 for one of the highest value property for the player, decreases the rent level by 1 for the highest value property held by the opponents. Note: Rent level cannot go lower than 1.

Players:

1. Player A
2. Player B
3. Player C

Properties:

1. Cochin {Value: 120, Color: Green, Rent per Level:100,160,260,440,860 }
2. Ooty {Value: 400, Color: Green, Rent per Level:300,400,560,810,1600}
3. Bombay {Value: 500, Color: Red, Rent per Level:400,520,680,900,1800}
4. Ahmedabad {Value: 300, Color: Red, Rent per Level:200,350,480,800,1200}
5. Chennai {Value: 700, Color: Blue, Rent per Level:600,900,1250,1500,1900}
6. Bangalore {Value: 450, Color: Blue, Rent per Level:300,400,560,810,1600}
7. Delhi {Value: 500, Color: Yellow, Rent per Level:400,520,680,900,1800}
8. Darjeeling {Value: 600, Color: Yellow, Rent per Level:400,700,1000,1150,1400}

As soon as a player buys a property, its rent level is set to 1. When another player lands on this property, he pays Rent amount corresponding to Level 1 (say $100 for Cochin), post which the rent level goes up to Level 2 (say $160 for Cochin).

Write a program that takes input values for Player and Property tile. Post the given set of inputs, the program prints out the state of all the players and properties on the board.

Inputs:

1. Player A lands on Bombay
2. Player B lands on Delhi
3. Player C lands on Chennai
4. Player A lands on Special
5. Player B lands on Bombay
6. Player C lands on Ooty
7. Player A lands on Ahmedabad
8. Player B lands on Darjeeling
9. Player C lands on Special
10. Player A lands on Delhi
11. Player B crosses GO
12. Player C lands on Bombay

Expected Outputs:

Players

- Player A - {Balance: $800}
- Player B - {Balance: $580}
- Player C - {Balance: $380}

Properties

- Cochin {Owner: None (For sale), Current Rent Level: N/A, Color: Green}
- Ooty {Owner: Player C, Current Rent Level:1, Color: Green}
- Bombay {Owner: Player A, Current Rent Level: 3, Color: Red}
- Ahmedabad {Owner: Player A, Color: Red}
- Chennai {Owner: Player C, Current Rent Level:1, Color: Blue}
- Bangalore {Owner: None (For sale), Current Rent Level:N/A, Color: Blue}
- Delhi {Owner: Player B, Current Rent Level: 2, Color: Yellow}
- Darjeeling {Owner: Player B, Current Rent Level: 1, Color: Yellow}

End of document