

Requirements

The first assignment is to produce a set of Requirements documents for the Monopoly system. These include the following:

- Top level use case diagram.
- Detail use case diagrams.
- Complete specifications for all use cases.
- Supplementary specification document (non-functional requirements and implementation constraints).
- Glossary.

Follow the official rules of Monopoly with the following changes:

- Don't worry about the rules concerning auctions or private transactions.
- Don't worry about Chance and Community Chest. Treat these spaces as "do nothing" spaces, just like GO, JustVisiting (jail), and FreeParking.
- Perform some money transactions automatically (see below).

You can assume that the system you will be specifying will have a user interface for all of the players on one machine. There will be no computer player. Don't worry about any of the GUI classes and functionality. You could even assume that all inputs and outputs are in the form of text messages if this would simplify things.

It is desirable that this system be highly customizable. The names of squares, graphics on the board, property prices and rents, kinds of improvements and their prices, forms of currency, contents of Chance and Community Chest cards (pictures, text, and functionality), etc. should be read in from an initialization file at the beginning of the game.

The official rules of the game should be followed, with the exception that all monies will be transferred automatically when due. The user should be notified by the system whenever automatic money actions are taken. Here are some of the automatic money transactions:

- Receive \$200 salary when passing GO.
- Transfer rent due when landing on someone else's property.
- Pay \$75 Luxury Tax.
- Pay Income tax – the lower of \$200 or 10% of net worth automatically computed by the system.
- Pay \$50 if you are in Jail and don't roll doubles on your third try.

Note: when there is insufficient cash on hand to complete one of these automatic payments, the system should determine whether the player would be able to raise the funds by selling houses or hotels or by mortgaging properties (i.e., whether the player's net worth is greater than the amount owed). If the system determines that the player can't pay even then, the system should declare the player bankrupt. If the system determines that the player can raise the money, the player should decide which actions should be taken to raise a sufficient amount to handle the debt.