README file – Ex3

Daniel Kuris

Danielkuris6@gmail.com

Catan Project

This project implements a simplified version of the board game "Catan." The main components are the Board, Land, Player, and Catan classes. Below is a detailed explanation of each class, their methods, and how they interact.

Classes and Methods

Board Class

The Board class represents the game board, which is composed of multiple Land objects.

Methods:

- **Board()**: Constructor that initializes the board.
- **Land& get_land(int index)**: Returns a reference to the land at the specified index.
- **Place robber and get robber location**: Places robber at certain land / returns the index of the land on which the robber is.

Land Class

The Land class represents individual lands on the board, each having a type and a probability number. This component has the details of the current board state. Each land saves the details on it, and knows its neighbours.

Attributes:

- **std::string type**: Type of the land (e.g., wood, brick, wool).
- **int probability**: Probability number (2-12, excluding 7).
- **Int number**: Index (0-18)
- **Vector<int> settlements**: Size 6, 0 if no settlement, player's color if occupied. Indexes the settlements on current land.
- **:vector<int> cities:** (0-6) indexes similarly to settlements
- **vector<int> roads:** (0-6) indexes similarly to settlements

Methods:

- **Land(std::string type,int number, int probability,)**: Constructor that initializes the land with a type and probability and index place on board.
- **std::string get_type() const**: Returns the type of the land.
- **int get_probability() const**: Returns the probability number of the land.

- **Occupy**: Each build (settlement,city,road) has an occupy method that validates the location and occupies current land's vector with player's color.
- **OccupyNeighbours**: Each build has occupy neighbours that occupies the neighbour lands that share the same edge / point with current land. This method uses symmetry to easily get the neighbours.
- **ValidateLocation**: Each build has validate method that makes sure , using Catan rules, that the building is correct.

Player Class

The Player class represents players in the game, each having resources and methods for trading and building.

Attributes:

- **std::map<std::string, int> resources**: Map of resources with their quantities.
- **int color**: Player index. Follows the actual catan game.
- **String name:** The username of player
- **vector<int> DevelopmentCards:** Size 5, represents the amount of each development card. [0] has the amount of development cards of type 0.

Methods:

- **Constructor Player::Player(const std::string &name, int color)**
    - Initializes a player with a given name and color.
    - Initializes points, lands, developmentCards, usedKnights, and resources.
- **const std::string& Player::getName() const**
    - Returns the player's name.
- **int Player::getColor() const**
    - Returns the player's color.
- **Resource Management Methods**
    - **bool Player::hasResources(const std::map<std::string, int> &cost) const**
        - Checks if the player has enough resources specified by cost.
    - **void Player::deductResources(const std::map<std::string, int> &cost)**
        - Deducts resources specified by cost from the player.
- **Placement and Upgrade Methods**
    - **void Player::initialOccupy(Catan &game)**
        - Handles the initial placement of settlements and roads for the player on the board.

    - **Placement methods:** Each building option has its placement method that gets index for land and spot on that land. Then validates the user has the resources for the placement. The method then uses that certain land's method to occupy it.

- **Turn and Dice Roll Methods**
    - **void Player::rollDice(Catan &game)**
        - Simulates a dice roll and handles resource collection based on the roll. Deals with rolling 7 by discarding, if needed, half the cards for each player that has over 7.
    - **void Player::endTurn(Catan &game)**
        - Ends the player's turn.
- **Trading Methods**
    - **void Player::trade(Player &other, const std::vector<std::string> &offer, const std::vector<int> &offerAmounts, const std::vector<std::string> &request, const std::vector<int> &requestAmounts)**
        - Facilitates resource trading between the player and other.
- **Development Card Methods**
    - Each development card has its use-method.
- **Miscellaneous**
    - **void Player::printPoints() const**
        - Prints the player's current points.

Catan (game) Class

The Catan class represents a simplified version of a Catan-like board game, encapsulating the game's state and logic.

Attributes

- **Board board**: Represents the game board where players build settlements, roads, and manage resources.
- **std::vector<Player> players**: Holds instances of Player representing participants in the game.
- **int currentPlayerIndex**: Tracks the index of the current player taking their turn.
- **int mostKnights**: Tracks the number of knights played by any single player.
- **int mostKnightsPlayer**: Stores the player ID who has played the most knights.

main.cpp

The main.cpp file serves as the entry point for the program, where the game is initialized and started.

Main Function:

- Includes complex class for complex numbers

- HandlePlayerActions method: Deals with the actions a player chooses via commands, including “ help, build, pass, buy, trade “ and more.
- Initializes the game (granting each land its type and probability, initializes players..)
- Runs the game until each player played a turn