Note 02: Inheritance & Polymorphism

Instruction

We are going to create multiplayer games using inheritance and polymorphism. We are provided the header file '`Multi.h`' that contains the interface named *Multi* that consists of the fundamental multiplayer game functionalities, namely

- `intro()` - provides the name of the game and any necessary instructions.
- `move(str)` - validates input and performs moves.
- `playable()` - checks if the game is playable.
- `won()` - checks if the current player won.
- `next()` - changes to the next player.
- `current()` - retrieves the identification of the current player.
- `toString()` - displays the game content.

and the C++ file '`main.cpp`' that executes the game.

Multiplayer games tend to follow the same pattern; players alternate turns performing a move until a player wins or the game is no longer playable. Hence, we can use inheritance and polymorphism to have '`main.cpp`' define the general multiplayer play function, `play()`, using an *Multi* reference object, and then plug and play multiplayer games by simply defining a concrete class that inherits *Multi* and invoking `play()` with our game object.

Tic-Tac-Toe

Try to create the game Tic-Tac-Toe in the following stages.

Stage 1: Create a header file named '`TiTaTo.h`' that contains a header guard and includes the libraries *iostream*, *string*, *cctype*, *sstream*, and '`Multi.h`'.

Stage 2: Within the namespace *dsn*, create an empty class named *TicTacToe* that publicly inherits *Multi*.

Stage 3: Within the class *TicTacToe*, declare a private string field name *board*, a private integer field named *player*, and a private static constant string field named *tokens*, and then initialize *tokens* to `"OX"`.

Stage 4: Within the class *TicTacToe*, define the special member functions such that the initial values of *board* and *player* are `"*********"` and 0, respectively.

Stage 5: Within the class *TicTacToe*, override `intro()` so that it return the string `"TIC-TAC-TOE"`.

Stage 6: Within the class *TicTacToe*, override `next()` so that it assigns (*player* plus 1) mod 2 to *player*.

Stage 7: Within the class *TicTacToe*, override `current()` so that it returns the string concatenation of `"Player "` and the string representing *player* plus 1.

Stage 8: Within the class *TicTacToe*, override `toString()` so that it displays

```
Player py:

0 | 1 | 2
3 | 4 | 5
6 | 7 | 8
```

where *py* is *player* plus 1, and *d* (where d is a digit) is the element of *board* with index *d*. If the character of **board** is `"*"` replace it with a space.

Stage 9: Within the class *TicTacToe*, override `move()` so that it assigns *player*'s token (the element of *token* whose index equals *player*) to the element of *board* whose index equals the input and then returns true only if the input represents a digit between 0 and 8, inclusively, and the element of *board* is current an asterisk; otherwise, it returns false.

Stage 10: Within the class *TicTacToe*, define the private Boolean constant methods `horizontal()`, `vertical()`, and `diagonal()` that take no parameters and check for a horizontal row win, vertical row win, and diagonal row win, respectively. Next, override `won()` so that it returns the conjunction of the three functions.

Stage 11: Within the class *TicTacToe*, define the private Boolean constant method `available()` that no parameters and returns true if *board* contains an asterisk; otherwise, it returns false. Next, override `playable` so that it returns the disjunction of `available()` and the negation of `won()`.

Stage 12: In '`main.cpp`', include the header file '`TiTaTo.h`'.

Stage 13: With the main function of '`main.cpp`', create a *TicTacToe* object and call `play()` with the object as the argument.