# Exercise 1: TicTacToe In Objects

Write a two-player tic-tac-toe program. Prompt the players one at a time for their move (a number 1 through 9 for squares left to right and top to bottom). Print the board after each move. Make sure moves are legal and detect a winning move and congratulate the winner!

Use objects for the Board and Player. Their interfaces should be:

class Board {

public:

/\*\*

\* This method clears the board in preparation for a new game.

\*/

void clear();

/\*\*

\* This method returns the contents of the given cell.

\* @param cellNumber the cell number [0..8]

\* @return the contents: 0 (empty), 1 (player1), 2 (player2)

\*/

int getCell(int cellNumber);

/\*\*

\* This method registers a players move on the board and returns

\* true if the move was a win.

\* @param cellNumber the cell number [0..8]

\* @return true if the move was a win

\*/

bool setCell(int cellNumber, bool isPlayer1);

/\*\*

\* This method prints the board to the screen in ASCII art.

\*/

void print();

};

class Player {

public:

/\*\*

\* This method identifies the player as Player1 or Player2

\* @param isPlayer1 true if player1 or false if player2

\*/

void setPlayerNumber(bool isPlayer1);

/\*\*

\* This method gets the players next valid move.

\* @param board the game board for move checking

\* @return the player's input [0..8]

\*/

int getMove(Board& board);

/\*\*

\* This method prints the "you won" message.

\*/

void printWin();

};

Suggestions:

1. Create “Board.h”, “Player.h”, “Board.cpp”, “Player.cpp”, and “Game.cpp” files. Use #ifdefs in the headers and #includes in the cp files.
2. Type in the given interfaces in the header files. Make stub methods in the cpp files and put print statements in them. Put test code in main to call all the object methods to test.
3. Flesh out the Board methods and data using the code from last time. Use test code in main to set some cells and print the board.
4. Flesh out the Player methods and data using the code from last time. The “playerNumber” is used to control the wording in prompts and messages.
5. Flesh out the game loop in main.

### Board.h

#ifndef BOARD\_H\_

#define BOARD\_H\_

class Board {

int cells[9];

// Helper functions

char Board::getPrintCharacter(int contents);

bool checkForWin();

public:

/\*\*

\* This method clears the board in preparation for a new game.

\*/

void clear();

/\*\*

\* This method returns the contents of the given cell.

\* @param cellNumber the cell number [0..8]

\* @return the contents: 0 (empty), 1 (player1), 2 (player2)

\*/

int getCell(int cellNumber);

/\*\*

\* This method registers a players move on the board and returns

\* true if the move was a win.

\* @param cellNumber the cell number [0..8]

\* @return true if the move was a win

\*/

bool setCell(int cellNumber, bool isPlayer1);

/\*\*

\* This method prints the board to the screen in ASCII art.

\*/

void print();

};

#endif

### Board.cpp

#include "Board.h"

#include <iostream>

using namespace std;

void Board::clear() {

for(int x=0;x<9;++x) {

cells[x] = 0;

}

}

int Board::getCell(int cellNumber) {

return cells[cellNumber];

}

bool Board::setCell(int cellNumber, bool isPlayer1) {

if(isPlayer1) {

cells[cellNumber] = 1;

} else {

cells[cellNumber] = 2;

}

return checkForWin();

}

char Board::getPrintCharacter(int contents) {

if(contents==0) return ' ';

if(contents==1) return 'X';

if(contents==2) return 'O';

return '?';

}

void Board::print() {

cout << getPrintCharacter(cells[0]) << " | "

<< getPrintCharacter(cells[1]) << " | "

<< getPrintCharacter(cells[2]) << endl;

cout << "---------" << endl;

cout << getPrintCharacter(cells[3]) << " | "

<< getPrintCharacter(cells[4]) << " | "

<< getPrintCharacter(cells[5]) << endl;

cout << "---------" << endl;

cout << getPrintCharacter(cells[6]) << " | "

<< getPrintCharacter(cells[7]) << " | "

<< getPrintCharacter(cells[8]) << endl;

}

bool Board::checkForWin() {

for(int p=1;p<3;++p) {

// Horizontal

if(cells[0]==p && cells[1]==p && cells[2]==p) return true;

if(cells[3]==p && cells[4]==p && cells[5]==p) return true;

if(cells[6]==p && cells[7]==p && cells[8]==p) return true;

// Vertical

if(cells[0]==p && cells[3]==p && cells[6]==p) return true;

if(cells[1]==p && cells[4]==p && cells[7]==p) return true;

if(cells[2]==p && cells[5]==p && cells[8]==p) return true;

// Diagonal

if(cells[0]==p && cells[4]==p && cells[8]==p) return true;

if(cells[2]==p && cells[4]==p && cells[6]==p) return true;

}

return false;

}

### Player.h

#ifndef PLAYER\_H\_

#define PLAYER\_H\_

#include "Board.h"

class Player {

bool player;

public:

/\*\*

\* This method identifies the player as Player1 or Player2

\* @param isPlayer1 true if player1 or false if player2

\*/

void setPlayerNumber(bool isPlayer1);

/\*\*

\* This method gets the players next valid move.

\* @param board the game board for move checking

\* @return the player's input [0..8]

\*/

int getMove(Board& board);

/\*\*

\* This method prints the "you won" message.

\*/

void printWin();

};

#endif

### Player.cpp

#include "Player.h"

#include <iostream>

using namespace std;

void Player::setPlayerNumber(bool isPlayer1) {

player = isPlayer1;

}

int Player::getMove(Board& board) {

int move;

while(true) {

if(player) {

cout << "Pick a square Player 1: ";

} else {

cout << "Pick a square Player 2: ";

}

cin >> move;

if(move>=1 && move<=9 && board.getCell(move-1)==0) {

return move-1; // Return 0 to 8

}

cout << "Invalid move" << endl;

}

}

void Player::printWin() {

if(player) {

cout << "You win player 1!" << endl;

} else {

cout << "You win player 1!" << endl;

}

}

### Game.cpp

#include <iostream>

using namespace std;

#include "Board.h"

#include "Player.h"

void main() {

Board board;

Player player1;

Player player2;

// TODO: The Board constructor should do this first time

board.clear();

// TODO: Would be better to pass to constructor

player1.setPlayerNumber(true);

player2.setPlayerNumber(false);

while(true) {

board.print();

int move = player1.getMove(board);

bool win = board.setCell(move,true); // TODO: Use player.isPlayer1()

if(win) {

player1.printWin();

break;

}

// TODO: Use an array of 2 players instead of separately

board.print();

move = player2.getMove(board);

win = board.setCell(move,false); // Use player.isPlayer2() instead

if(win) {

player2.printWin();

break;

}

}

system("pause");

}