```cpp
#include "Player.h"
#include "Card.h"
#include "RatATat.h"

#define _CRTDBG_MAP_ALLOC
#include <crtdbg.h>
#ifdef _DEBUG
#ifndef DBG_NEW
#define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
#define new DBG_NEW
#endif
#endif // _DEBUG

Player::Player(string name, RatATat* pointer_to_game_ratatat)
{
    this->m_player_name = name;
    this->m_player_pointer_to_ratatat = pointer_to_game_ratatat;
    m_player_card_arr = new Card * [m_player_number_of_cards];
    memset(m_player_card_arr, NULL, m_player_number_of_cards * sizeof(Card*));

    this->m_player_cards_sum = 0;
    this->m_indx_to_put_a_card_in_player_hand = 0;
}

Player::~Player()
{
    for (int i = 0; i < m_player_number_of_cards; i++)
        delete m_player_card_arr[i];
    delete[] m_player_card_arr;
}

ostream& operator<<(ostream& os, const Player& player)
{
    os << player.m_player_name << ": " << endl;
    os << *player.m_player_card_arr[0] << " , ";
    os << "hidden card , hidden card , ";
    os << *player.m_player_card_arr[3] << endl;
    return os;
}

//////////////////////////////
//"Extras":

string Player::get_player_name()
{
    return this->m_player_name;
}

unsigned int Player::get_player_cards_sum() const
{
    return this->m_player_cards_sum;
}
```

```cpp

int Player::get_player_min_card_indx()
{
    return m_player_min_card_indx;
}

int Player::get_player_max_card_indx()
{
    return m_player_max_card_indx;
}

void Player::print_msg_choose_card_indx() const
{
    cout << "which card do you want want to choose? (0-3)" << endl;
}

void Player::put_card_to_player_hand(Card* card)
{
    m_player_card_arr[m_indx_to_put_a_card_in_player_hand] = card;
    m_indx_to_put_a_card_in_player_hand++;
}

void Player::calc_sum_of_player_hand()
{
    for (int i = 0; i < m_player_number_of_cards; i++)
        m_player_cards_sum = m_player_cards_sum + m_player_card_arr[i]->get_card_value();
}

void Player::show_player_hand_and_make_it_valid_to_calc_sum()
{
    cout << m_player_name << ": " << endl;
    for (int i = 0; i < m_player_number_of_cards; i++)
    {
        cout << *m_player_card_arr[i] << " , ";
        //here need to make hand valid to calc sum , then show hand again:
        while (m_player_card_arr[i]->is_special_card())
        {
            m_player_pointer_to_ratatat->throw_card_to_discard_pile(m_player_card_arr[i]);
            m_player_card_arr[i] = m_player_pointer_to_ratatat->get_card_from_unused_pile();
        }
    }
    cout << endl;
    for (int i = 0; i < m_player_number_of_cards; i++)
        cout << *m_player_card_arr[i] << " , ";
    cout << endl;
}

Card* Player::get_card_pointer_by_indx(int indx)
{
    return m_player_card_arr[indx];
```

```
}

//void Player::show_card_by_indx(int indx)
//{
//  cout << *m_player_card_arr[indx];
//}

void Player::replace_cards(int indx_of_card_to_replace, Card*& card_b_pointer)
{
    Card* tmp_card_pointer = m_player_card_arr[indx_of_card_to_replace];
    m_player_card_arr[indx_of_card_to_replace] = card_b_pointer;
    card_b_pointer = tmp_card_pointer;
    return;
}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//class AI_player:

int AI_player::ChooseOption_from_card_menu(Card& currentCard)
{
    int max_option = currentCard.get_card_action_menu_max_num();
    int rand_option = 1 + rand() % max_option;
    return rand_option;
}

int AI_player::choose_card_indx()                    //for play cards
{
    print_msg_choose_card_indx();
    int rand_option = rand() % m_player_max_card_indx;
    cout << rand_option << endl;
    return rand_option;
}

int AI_player::choose_player(Player** players_arr, int curr_player)  //for swap!
{
    cout << "with which player do you want to swap card with?" << endl;
    int player_answer_which_player_to_swap_with = rand() % m_player_pointer_to_ratatat->get_number_of_players();
    while (player_answer_which_player_to_swap_with == curr_player)
        player_answer_which_player_to_swap_with = rand() % m_player_pointer_to_ratatat->get_number_of_players();
    cout << player_answer_which_player_to_swap_with << endl;
    return player_answer_which_player_to_swap_with;
}

int AI_player::player_choice_from_what_pile_to_pick(int number_of_option)
{
    int ans(1 + rand() % number_of_option);
    cout << ans << endl;
```

```cpp
    return ans;
}

int AI_player::player_answer_to_call_ratatat(int number_of_option)
{
    int ans = (1 + rand() % number_of_option);
    cout << ans << endl;
    return ans;
}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//class Human_player:

int Human_player::ChooseOption_from_card_menu(Card& currentCard)
{
    int choice;
    int max_option = currentCard.get_card_action_menu_max_num();
    cin >> choice;
    while (choice < 1 || choice > max_option)
        cin >> choice;
    return choice;
}

int Human_player::choose_card_indx()
{
    int player_answer_which_card_to_choose;
    print_msg_choose_card_indx();
    cin >> player_answer_which_card_to_choose;
    while (player_answer_which_card_to_choose < Player::get_player_min_card_indx() || player_answer_which_card_to_choose > Player::get_player_max_card_indx())
        cin >> player_answer_which_card_to_choose;
    return player_answer_which_card_to_choose;
}

int Human_player::choose_player(Player** players_arr, int curr_player)
{
    int player_answer_which_player_to_swap_with;
    int number_of_players = m_player_pointer_to_ratatat->get_number_of_players();
    cout << "with which player do you want to swap card with?" << endl;
    int i;
    for (i = 0; i < number_of_players; i++)
        cout << i << ") " << players_arr[i]->get_player_name() << " , ";
    cout << endl;
    cin >> player_answer_which_player_to_swap_with;
    while (player_answer_which_player_to_swap_with < 0 || player_answer_which_player_to_swap_with >= number_of_players || player_answer_which_player_to_swap_with == curr_player)
        cin >> player_answer_which_player_to_swap_with;

    return player_answer_which_player_to_swap_with;
}

int Human_player::player_choice_from_what_pile_to_pick(int number_of_option)
{
    int answer;
    cin >> answer;
    while (answer < 1 || answer >number_of_option)
        cin >> answer;
    return answer;
}

int Human_player::player_answer_to_call_ratatat(int number_of_option)
{
    int answer;
    cin >> answer;
    while (answer < 1 || answer > number_of_option)
        cin >> answer;
    return answer;
}

```