```
#include "Card.h"
#include "Player.h"
#include "Pile.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

//initialize static members:
int Card::m_number_of_cards_already_made(0);
Card::Card(const string& card_type, const int card_value)
{
    m_text = card_type ;
    m_value = card_value;
    m_number_of_cards_already_made++;
}

ostream& operator<<(ostream& os, const Card& card)
{
    os << card.m_text ;
    return os;
}

//Extras:
int Card::get_number_of_cards_already_made()
{
    return m_number_of_cards_already_made;
}

int Card::get_total_number_of_cards_in_1_ratatat_pile()
{
    return m_total_number_of_cards_in_1_ratatat_pile;
}

int Card::get_card_value() const
{
    return m_value;
}

bool Card::is_special_card() const
{
    if (m_value == -1)
        return true;
    return false;
}

////////////////////////////////////////////////////////////////////////// ↩
```

```cpp
    /////////////////////////////////////////
//class Play_card:

int Play_card::m_Play_cards_hist_total[10] = { 4,4,4,4,4,4,4,4,4,9 };
int Play_card::m_Play_cards_hist_made_till_now[10] = { 0 };
Play_card::Play_card(const string& card_type, const int card_value) : Card
  (card_type, card_value)
{
    m_Play_cards_hist_made_till_now[card_value]++;
}

//the card virtual methodes:
void Play_card::use(Player** players, int curr_player, RatATat& rat)
{
    int player_answer_which_card_to_replace = players[curr_player]-
      >choose_card_indx();
    Card* my_card_to_swap = players[curr_player]->get_card_pointer_by_indx
      (player_answer_which_card_to_replace);
    Card* tmp_pointer_to_this = this;

    players[curr_player]->replace_cards
      (player_answer_which_card_to_replace, tmp_pointer_to_this);
    rat.throw_card_to_discard_pile(my_card_to_swap);
}

void Play_card::print_card_action_menu() const
{
    cout << "Choose option:" << endl;
    cout << "1. Discard" << endl;
    cout << "2. Replace with one of my cards" << endl;
}

int Play_card::get_card_action_menu_max_num() const
{
    return 2;
}

/////////////////////////////////////////////////////////////////////
//class Rat_card:
unsigned int Rat_card::m_num_cards(0);
Rat_card::Rat_card(unsigned int card_value) :Play_card("Rat," + to_string
  (card_value), card_value)
{
    m_num_cards++;
}

//static methodes:
int Rat_card::get_total_cards()
{
    return m_num_cards;
}

int Rat_card::toss_val()
```

```cpp
{
    int rand_card_number;
    rand_card_number = m_Rat_card_min_value + rand() % 
      m_number_of_Rat_card_types;
    while (m_Play_cards_hist_made_till_now[rand_card_number] == 
      m_Play_cards_hist_total[rand_card_number])
    {
        rand_card_number = m_Rat_card_min_value + rand() % 
          m_number_of_Rat_card_types;
    }
    return rand_card_number;
}

//Extras:
int Rat_card::get_total_Rat_cards_in_1_pile()
{
    return m_total_Rat_cards_in_1_pile;
}


/////////////////////////////////////////////////////////////////////
//class Cat_card:
unsigned int Cat_card::m_num_cards(0);
Cat_card::Cat_card(unsigned int card_value) :Play_card("Cat," + to_string 
  (card_value), card_value)
{
    m_num_cards++;
}

//static methodes:
int Cat_card::get_total_cards()
{
    return m_num_cards;
}

int Cat_card::toss_val()
{
    int rand_card_number;
    rand_card_number = m_Cat_card_min_value + rand() % 
      m_number_of_Cat_card_types;
    while (m_Play_cards_hist_made_till_now[rand_card_number] == 
      m_Play_cards_hist_total[rand_card_number])
    {
        rand_card_number = m_Cat_card_min_value + rand() % 
          m_number_of_Cat_card_types;
    }
    return rand_card_number;
}

//Extras:
int Cat_card::get_total_Cat_cards_in_1_pile()
{
    return m_total_Cat_cards_in_1_pile;
}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//class Special_card :

Special_card::Special_card(const string& special_card_type , const int card_value) : Card(special_card_type, card_value) {}

void Special_card::print_card_action_menu() const
{
    cout << "Choose option:" << endl;
    cout << "1. Discard without use" << endl;
    cout << "2. Use special card" << endl;
}

int Special_card::get_card_action_menu_max_num() const
{
    return 2;
}

/////////////////////////////////////////////////////////////////////////
//class Draw2_card:
unsigned int Draw2_card::m_num_cards(0);
Draw2_card::Draw2_card() : Special_card("Draw 2")
{
    m_num_cards++;
}

//the card virtual methodes:
void Draw2_card::use(Player** players, int curr_player, RatATat& rat)
{
    rat.play_turn();
    rat.play_turn();
    rat.throw_card_to_discard_pile(this);
}

//static methodes:
int Draw2_card::get_total_cards()
{
    return m_num_cards;
}

//Extras:
int Draw2_card::get_total_Draw2_cards_in_1_pile()
{
    return m_total_Draw2_cards_in_1_pile;
}

////////////////////////////////////////////////////////////////////////////
//class Swap_card:
unsigned int Swap_card::m_num_cards(0);
```

```cpp
Swap_card::Swap_card() : Special_card("Swap")
{
    m_num_cards++;
}

//the card virtual methodes:
void Swap_card::use(Player** players, int curr_player, RatATat& rat)
{
    int player_answer_which_player_to_swap_with = players[curr_player]->choose_player(players, curr_player);
    int player_answer_which_one_of_his_card_to_replace = players[curr_player]->choose_card_indx();
    int player_answer_which_other_player_card_to_replace = players[curr_player]->choose_card_indx();
    Card* my_card = players[curr_player]->get_card_pointer_by_indx(player_answer_which_one_of_his_card_to_replace);
    Card* his_card = players[player_answer_which_player_to_swap_with]->get_card_pointer_by_indx(player_answer_which_other_player_card_to_replace);
    swap_cards(my_card, his_card);

    rat.throw_card_to_discard_pile(this);
}

//static methodes:
int Swap_card::get_total_cards()
{
    return m_num_cards;
}

//Extras:
int Swap_card::get_total_Swap_cards_in_1_pile()
{
    return m_total_Swap_cards_in_1_pile;
}

void Swap_card::swap_cards(Card*& card_a_pointer, Card*& card_b_pointer)
{
    Card* tmp_card_pointer = card_a_pointer;
    card_a_pointer = card_b_pointer;
    card_b_pointer = tmp_card_pointer;
    return;
}

/////////////////////////////////////////////////////////////////////////////
//class Peek_card:
unsigned int Peek_card::m_num_cards(0);
Peek_card::Peek_card() : Special_card("Peek")
{
    m_num_cards++;
}

```

```cpp
//the card virtual methodes:
void Peek_card::use(Player** players, int curr_player, RatATat& rat)
{

    int player_answer_which_one_of_his_card_to_peek = players[curr_player]->choose_card_indx();
    cout << *players[curr_player]->get_card_pointer_by_indx(player_answer_which_one_of_his_card_to_peek) << endl;

    rat.throw_card_to_discard_pile(this);
}

//static methodes:
int Peek_card::get_total_cards()
{
    return m_num_cards;
}

//Extras:
int Peek_card::get_total_Peek_cards_in_1_pile()
{
    return m_total_Peek_cards_in_1_pile;
}

/////////////////////////////////////////////////////////////////////////////////////////////////

```