

```
1 #include "Card.h"
2 #include "Player.h"
3 #include "Pile.h"
4 #include "RatATat.h"
5
6 #define _CRTDBG_MAP_ALLOC
7 #include <crtdbg.h>
8 #ifdef _DEBUG
9 #ifndef DBG_NEW
10 #define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
11 #define new DBG_NEW
12 #endif
13 #endif // _DEBUG
14
15 //initialize static members:
16 int Card::m_number_of_cards_already_made(0);
17 Card::Card(const string& card_type, const int card_value)
18 {
19     m_text = card_type ;
20     m_value = card_value;
21     m_number_of_cards_already_made++;
22 }
23
24 ostream& operator<<(ostream& os, const Card& card)
25 {
26     os << card.m_text ;
27     return os;
28 }
29
30 //Extras:
31 int Card::get_number_of_cards_already_made()
32 {
33     return m_number_of_cards_already_made;
34 }
35
36 int Card::get_total_number_of_cards_in_1_ratatat_pile()
37 {
38     return m_total_number_of_cards_in_1_ratatat_pile;
39 }
40
41 int Card::get_card_value() const
42 {
43     return m_value;
44 }
45
46 bool Card::is_special_card() const
47 {
48     if (m_value == -1)
49         return true;
50     return false;
51 }
52
53 //////////////////////////////////////////////////
```

```

////////////////////////////////////////
54 //class Play_card:
55
56 int Play_card::m_Play_cards_hist_total[10] = { 4,4,4,4,4,4,4,4,4,9 };
57 int Play_card::m_Play_cards_hist_made_till_now[10] = { 0 };
58 Play_card::Play_card(const string& card_type, const int card_value) : Card
    (card_type, card_value)
59 {
60     m_Play_cards_hist_made_till_now[card_value]++;
61 }
62
63 //the card virtual methodes:
64 void Play_card::use(Player** players, int curr_player, RatATat& rat)
65 {
66     int player_answer_which_card_to_replace = players[curr_player]-
        >choose_card_indx();
67     Card* my_card_to_swap = players[curr_player]->get_card_pointer_by_indx
        (player_answer_which_card_to_replace);
68     Card* tmp_pointer_to_this = this;
69
70     players[curr_player]->replace_cards
        (player_answer_which_card_to_replace, tmp_pointer_to_this);
71     rat.throw_card_to_discard_pile(my_card_to_swap);
72 }
73
74 void Play_card::print_card_action_menu() const
75 {
76     cout << "Choose option:" << endl;
77     cout << "1. Discard" << endl;
78     cout << "2. Replace with one of my cards" << endl;
79 }
80
81 int Play_card::get_card_action_menu_max_num() const
82 {
83     return 2;
84 }
85
86 //////////////////////////////////////////
87 //class Rat_card:
88 unsigned int Rat_card::m_num_cards(0);
89 Rat_card::Rat_card(unsigned int card_value) :Play_card("Rat," + to_string
    (card_value), card_value)
90 {
91     m_num_cards++;
92 }
93
94 //static methodes:
95 int Rat_card::get_total_cards()
96 {
97     return m_num_cards;
98 }
99
100 int Rat_card::toss_val()

```

```
101 {
102     int rand_card_number;
103     rand_card_number = m_Rat_card_min_value + rand() %
        m_number_of_Rat_card_types;
104     while (m_Play_cards_hist_made_till_now[rand_card_number] ==
        m_Play_cards_hist_total[rand_card_number])
105     {
106         rand_card_number = m_Rat_card_min_value + rand() %
            m_number_of_Rat_card_types;
107     }
108     return rand_card_number;
109 }
110
111 //Extras:
112 int Rat_card::get_total_Rat_cards_in_1_pile()
113 {
114     return m_total_Rat_cards_in_1_pile;
115 }
116
117
118 ///////////////////////////////////////////////////
119 //class Cat_card:
120 unsigned int Cat_card::m_num_cards(0);
121 Cat_card::Cat_card(unsigned int card_value) :Play_card("Cat," + to_string
    (card_value), card_value)
122 {
123     m_num_cards++;
124 }
125
126 //static methodes:
127 int Cat_card::get_total_cards()
128 {
129     return m_num_cards;
130 }
131
132 int Cat_card::toss_val()
133 {
134     int rand_card_number;
135     rand_card_number = m_Cat_card_min_value + rand() %
        m_number_of_Cat_card_types;
136     while (m_Play_cards_hist_made_till_now[rand_card_number] ==
        m_Play_cards_hist_total[rand_card_number])
137     {
138         rand_card_number = m_Cat_card_min_value + rand() %
            m_number_of_Cat_card_types;
139     }
140     return rand_card_number;
141 }
142
143 //Extras:
144 int Cat_card::get_total_Cat_cards_in_1_pile()
145 {
146     return m_total_Cat_cards_in_1_pile;
```

```

147 }
148
149 ////////////////////////////////////////////////// ➤
150 //////////////////////////////////////////////////
151 //class Special_card :
152
153 Special_card::Special_card(const string& special_card_type , const int ➤
154     card_value) : Card(special_card_type, card_value) {}
155
156 void Special_card::print_card_action_menu() const
157 {
158     cout << "Choose option:" << endl;
159     cout << "1. Discard without use" << endl;
160     cout << "2. Use special card" << endl;
161 }
162
163 int Special_card::get_card_action_menu_max_num() const
164 {
165     return 2;
166 }
167
168 ////////////////////////////////////////////////// ➤
169 //class Draw2_card:
170 unsigned int Draw2_card::m_num_cards(0);
171 Draw2_card::Draw2_card() : Special_card("Draw 2")
172 {
173     m_num_cards++;
174 }
175
176 //the card virtual methodes:
177 void Draw2_card::use(Player** players, int curr_player, RatATat& rat)
178 {
179     rat.play_turn();
180     rat.play_turn();
181     rat.throw_card_to_discard_pile(this);
182 }
183
184 //static methodes:
185 int Draw2_card::get_total_cards()
186 {
187     return m_num_cards;
188 }
189
190 //Extras:
191 int Draw2_card::get_total_Draw2_cards_in_1_pile()
192 {
193     return m_total_Draw2_cards_in_1_pile;
194 }
195
196 ////////////////////////////////////////////////// ➤
197 /
198 //class Swap_card:
199 unsigned int Swap_card::m_num_cards(0);

```

```

197 Swap_card::Swap_card() : Special_card("Swap")
198 {
199     m_num_cards++;
200 }
201
202 //the card virtual methodes:
203 void Swap_card::use(Player** players, int curr_player, RatATat& rat)
204 {
205     int player_answer_which_player_to_swap_with = players[curr_player]-
        >choose_player(players, curr_player);
206     int player_answer_which_one_of_his_card_to_replace = players
        [curr_player]->choose_card_indx();
207     int player_answer_which_other_player_card_to_replace = players
        [curr_player]->choose_card_indx();
208     Card* my_card = players[curr_player]->get_card_pointer_by_indx
        (player_answer_which_one_of_his_card_to_replace);
209     Card* his_card = players[player_answer_which_player_to_swap_with] -
        >get_card_pointer_by_indx
        (player_answer_which_other_player_card_to_replace);
210     swap_cards(my_card, his_card);
211
212     rat.throw_card_to_discard_pile(this);
213 }
214
215 //static methodes:
216 int Swap_card::get_total_cards()
217 {
218     return m_num_cards;
219 }
220
221 //Extras:
222 int Swap_card::get_total_Swap_cards_in_1_pile()
223 {
224     return m_total_Swap_cards_in_1_pile;
225 }
226
227 void Swap_card::swap_cards(Card*& card_a_pointer, Card*& card_b_pointer)
228 {
229     Card* tmp_card_pointer = card_a_pointer;
230     card_a_pointer = card_b_pointer;
231     card_b_pointer = tmp_card_pointer;
232     return;
233 }
234
235 ////////////////////////////////////////////////////
    ///
236 //class Peek_card:
237 unsigned int Peek_card::m_num_cards(0);
238 Peek_card::Peek_card() : Special_card("Peek")
239 {
240     m_num_cards++;
241 }
242

```

```
243 //the card virtual methodes:
244 void Peek_card::use(Player** players, int curr_player, RatATat& rat)
245 {
246
247     int player_answer_which_one_of_his_card_to_peek = players
        [curr_player]->choose_card_indx();
248     cout << *players[curr_player]->get_card_pointer_by_indx
        (player_answer_which_one_of_his_card_to_peek) << endl;
249
250     rat.throw_card_to_discard_pile(this);
251 }
252
253 //static methodes:
254 int Peek_card::get_total_cards()
255 {
256     return m_num_cards;
257 }
258
259 //Extras:
260 int Peek_card::get_total_Peek_cards_in_1_pile()
261 {
262     return m_total_Peek_cards_in_1_pile;
263 }
264
265 ////////////////////////////////////////
    ////////////////////////////////////////
266
```