

601.220 Intermediate Programming

Customized exceptions

Exceptions

```
// exceptions.cpp
1  #include <iostream>
2  #include <stdexcept>
3
4  using std::cout; using std::endl;
5  using std::runtime_error;
6  using std::out_of_range;
7
8  int main() {
9      try {
10         throw out_of_range("not a runtime_error");
11
12         cout << "no exception" << endl;
13     } catch(const runtime_error& e) {
14         cout << "runtime_error: " << e.what()
15         << endl;
16     } catch(const std::exception& e) {
17         // out_of_range is derived from exception
18         // but not from runtime_error
19         cout << "exception: " << e.what() << endl;
20     }
21     return 0;
22 }
```

```
$ g++ -o exceptions exceptions.cpp -std=c++11 -pedantic -Wall -Wextra
$ ./exceptions
exception: not a runtime_error
```

- C++ passes control to **first** `catch` block whose type equals or is a base class of the thrown exception
- Arrange catch blocks from **most to least specific** type

Customized Exceptions

You can define your own exception class, derived from `std::exception`

Since exceptions are related through inheritance, you can choose whether to catch a base class (thereby catching more different things) or a derived class

Here, we use a card-game example to demonstrate both points

Customized Exceptions: card_game.h

```
// card_game.h

1  #ifndef CARD_GAME_H
2  #define CARD_GAME_H
3
4  #include <iostream>
5  #include <sstream>
6  #include <stdexcept>
7  #include <vector>
8  #include <utility>
9  #include <string>
10 #include <algorithm>
11
12 enum class Suit { HEART, DIAMOND, SPADE, CLUB };
13 enum class Rank { ACE = 1, TWO, THREE, FOUR, FIVE,
14                 SIX, SEVEN, EIGHT, NINE, TEN,
15                 JACK, QUEEN, KING };
16
17 typedef std::pair<Suit, Rank> Card; //suit + rank
18
19 class BadCardError : public std::runtime_error {
20 public:
21     BadCardError(Card c) :
22         std::runtime_error("bad card"), card(c) {}
23 private:
24     Card card;
25 };

27 class CardGame {
28 public:
29     CardGame() : deck(), discard_pile() {
30         for(int s = (int)Suit::HEART;
31             s <= (int)Suit::CLUB; s++) {
32             for(int r = (int)Rank::ACE;
33                 r <= (int)Rank::KING; r++) {
34                 deck.push_back(std::make_pair(
35                     (Suit)s, (Rank)r));
36             }
37         }
38         std::random_shuffle(deck.begin(),
39                             deck.end());
40     }
41
42     Card draw();
43     void discard(Card c);
44     size_t deck_size() const {
45         return deck.size();
46     }
47
48 private:
49     std::vector<Card> deck, discard_pile;
50 };
51
52 #endif // CARD_GAME_H
```

Customized Exceptions: card_game.cpp

```
// card_game.cpp

1  #include "card_game.h"
2
3  Card CardGame::draw() {
4      Card c = deck.back();
5      deck.pop_back();
6      return c;
7  }
8
9  void CardGame::discard(Card c) {
10     // sanity check the card first
11     if(c.first < Suit::HEART || c.first > Suit::CLUB ||
12        c.second < Rank::ACE || c.second > Rank::KING)
13     {
14         throw BadCardError(c);
15     }
16     discard_pile.push_back(c);
17 }
```

Customized Exceptions: card_game_main1.cpp

```
// card_game_main1.cpp
```

```
1  #include "card_game.h"
2
3  using std::cout;  using std::endl;
4  int main() {
5      CardGame cg;
6      Card c = cg.draw();
7      try {
8          cg.discard(c);
9          cout << "no exception" << endl;
10     } catch(const std::runtime_error& e) {
11         cout << "runtime_error: " << e.what() << endl;
12     }
13     return 0;
14 }
```

```
$ g++ -o card_game_main1 card_game_main1.cpp card_game.cpp -std=c++11 -pedantic
```

```
$ ./card_game_main1
```

```
no exception
```

Customized Exceptions: card_game_main2.cpp

```
// card_game_main2.cpp
1  #include "card_game.h"
2
3  using std::cout; using std::endl;
4  int main() {
5      CardGame cg;
6      Card c = cg.draw();
7      try {
8          c.first = (Suit)5; // Card is now malformed!
9          cg.discard(c);
10         cout << "no exception" << endl;
11     } catch(const std::runtime_error& e) {
12         cout << "runtime_error: " << e.what() << endl;
13     }
14     return 0;
15 }
```

```
$ g++ -o card_game_main2 card_game_main2.cpp card_game.cpp -std=c++11 -pedantic
```

```
$ ./card_game_main2
```

```
runtime_error: bad card
```

Customized Exceptions: card_game_main3.cpp

```
// card_game_main3.cpp
1  #include "card_game.h"
2
3  using std::cout;  using std::endl;
4  int main() {
5      CardGame cg;
6      Card c = cg.draw();
7      try {
8          c.first = (Suit)5; // Card is now malformed!
9          cg.discard(c);
10         cout << "no exception" << endl;
11     } catch(const std::runtime_error& e) { // first specific
12         cout << "runtime_error: " << e.what() << endl;
13     } catch(const std::exception& e) { // then general
14         cout << "exception: " << e.what() << endl;
15     }
16     return 0;
17 }

$ g++ -o card_game_main3 card_game_main3.cpp card_game.cpp -std=c++11 -pedantic
$ ./card_game_main3
runtime_error: bad card
```