601.220 Intermediate Programming

Customized exceptions

Exceptions

```
// exceptions.cpp

#include <iostream>
#include <stdexcept>

using std::cout;  using std::endl;
using std::runtime_error;
using std::out_of_range;

int main() {
  try {
    throw out_of_range("not a runtime_error");
    cout << "no exception" << endl;
  } catch(const runtime_error& e) {
    cout << "runtime_error: " << e.what()
    << endl;
  } catch(const std::exception& e) {
    // out_of_range is derived from exception
    // but *not* from runtime_error
    cout << "exception: " << e.what() << endl;
  }
  return 0;
```

```
$ 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

#ifndef CARD_GAME_H
#define CARD_GAME_H

#include <iostream>
#include <sstream>
#include <stdexcept>
#include <vector>
#include <utility>
#include <string>
#include <algorithm>

enum class Suit { HEART, DIAMOND, SPADE, CLUB };
enum class Rank { ACE = 1, TWO, THREE, FOUR, FIVE,
                  SIX, SEVEN, EIGHT, NINE, TEN,
                  JACK, QUEEN, KING };

typedef std::pair<Suit, Rank> Card; //suit + rank

class BadCardError : public std::runtime_error {
public:
    BadCardError(Card c) :
    std::runtime_error("bad card"), card(c) { }
private:
    Card card;
};
```

```
class CardGame {
public:
    CardGame() : deck(), discard_pile() {
        for(int s = (int)Suit::HEART;
          s <= (int)Suit::CLUB; s++) {
            for(int r = (int)Rank::ACE;
              r <= (int)Rank::KING; r++) {
                deck.push_back(std::make_pair(
                  (Suit)s, (Rank)r));
            }
        }
        std::random_shuffle(deck.begin(),
          deck.end());
    }

    Card draw();
    void discard(Card c);
    size_t deck_size() const {
        return deck.size();
    }

private:
    std::vector<Card> deck, discard_pile;
};

#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

#include "card_game.h"

using std::cout;  using std::endl;
int main() {
    CardGame cg;
    Card c = cg.draw();
    try {
        cg.discard(c);
        cout << "no exception" << endl;
    } catch(const std::runtime_error& e) {
        cout << "runtime_error: " << e.what() << endl;
    }
    return 0;
}
```

```
$ 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

#include "card_game.h"

using std::cout;  using std::endl;
int main() {
    CardGame cg;
    Card c = cg.draw();
    try {
        c.first = (Suit)5; // Card is now malformed!
        cg.discard(c);
        cout << "no exception" << endl;
    } catch(const std::runtime_error& e) {
        cout << "runtime_error: " << e.what() << endl;
    }
    return 0;
}
```

```
$ 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
```