601.220 Intermediate Programming

Customized exceptions

Exceptions

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

- E.g. `catch(const std::runtime_error& e)` before `catch(const std::exception& e)`

Exceptions

```cpp
// exc_spec.cpp:
#include <iostream>
#include <stdexcept>

using std::cout;  using std::endl;

int main() {
    try {
        throw std::out_of_range("not a runtime_error");
        cout << "no exception" << endl;
    } catch(const std::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;
}
```

Exceptions

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

Less specific `catch(const std::exception& e)` block is used

Exceptions

You can define your own exception class, derived from `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

Following card-game example demonstrates both points

Exceptions: card_game.h

```cpp
// 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;
};
```

```cpp
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
```

Exceptions: `card_game.cpp`

```
// card_game.cpp:
#include "card_game.h"

Card CardGame::draw() {
    Card c = deck.back();
    deck.pop_back();
    return c;
}

void CardGame::discard(Card c) {
    // sanity check the card first
    if(c.first < Suit::HEART || c.first > Suit::CLUB ||
       c.second < Rank::ACE || c.second > Rank::KING)
    {
        throw BadCardError(c);
    }
    discard_pile.push_back(c);
}
```

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;
}
```

Exceptions

```
$ g++ -o card_game_main1 card_game_main1.cpp card_game.cpp
$ ./card_game_main1
no exception
```

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;
}
```

Exceptions

```
$ g++ -o card_game_main2 card_game_main2.cpp card_game.cpp
$ ./card_game_main2
runtime_error: bad card
```

Our `catch` block caught the exception

Exceptions: `card_game_main3.cpp`

```cpp
// card_game_main3.cpp:
#include "card_game.h"

using std::cout;  using std::endl;

int main() {
    CardGame cg;
    Card c = cg.draw();
    try {
        c.first = (Suit)5;
        cg.discard(c);
        cout << "no exception" << endl;
    } catch(const std::runtime_error& e) {
        // first catch block that either equals or is a
        // base class of the thrown exception is the one
        // used
        cout << "runtime_error: " << e.what() << endl;
    } catch(const std::exception& e) {
        cout << "exception: " << e.what() << endl;
    }
    return 0;
}
```

Exceptions

```
$ g++ -o card_game_main3 card_game_main3.cpp card_game.cpp
$ ./card_game_main3
runtime_error: bad card
```