Лабораторная работа №6.

Текст программы к заданию 2:

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13  14.  15.  16.  17.  18.  19.  20.  21.  22.  23.  24.  25.  26.  27.  28.  29.  30.  31  32..  33.  34.  35.  36.  37.  38.  39.  40.  41.  42.  43.  44.  45.  46.  47.  48.  49  50.  51.  52.  53.  54.  55.  56.  57.  58.  59.  60.  61.  62.  63.  64.  65.  66.  67.  68  69.  70.  71.  72.  73.  74.  75.  76.  77.  78.  79.  80.  81.  82.  83.  84.  85.  86.  87.  88.  89.  90.  91.  92.  93.  94.  95.  96.  97.  98.  99.  100.  101.  102. | #include <iostream>  #include <string>  #include <vector>  struct Book {  unsigned long int \_URN;  std::string \_title;  std::string \_author;  unsigned int \_year;  unsigned int \_count;  Book() {}  Book(unsigned long int URN, std::string title, std::string author, unsigned int year, unsigned int count) {  \_URN = URN;  if (title.length() <= 50) {  \_title = title;  } else {  std::cout << "Title is too long";  throw new std::exception();  }  if (author.length() <= 20) {  \_author = author;  } else {  std::cout << "Author's name is too long";  throw new std::exception();  }  \_year = year;  \_count = count;  }  int input() {  std::cout << "Input book's united registration number:\n";  std::cin >> \_URN;  std::cout << "Input book's title:\n";  std::cin.ignore();  std::getline(std::cin, \_title, '\n');  if (\_title.length() > 50) {  std::cout << "Error, try again!\n";  return -1;  }  std::cout << "Input book's author:\n";  std::getline(std::cin, \_author, '\n');  if (\_author.length() > 20) {  std::cout << "Error, try again!\n";  return -1;  }  std::cout << "Input book's year:\n";  std::cin >> \_year;  std::cout << "Input book's count:\n";  std::cin >> \_count;  return 0;  }  void output() {  std::cout << "Current URN: " << \_URN << '\n';  std::cout << "Current title: " << \_title << '\n';  std::cout << "Current author: " << \_author << '\n';  std::cout << "Current year: " << \_year << '\n';  std::cout << "Current count: " << \_count << '\n';  }  };  struct Library {  std::vector<Book> \_container;  int \_N;  Library() {}  Library (int N) {  \_container = std::vector<Book>(N);  for (int i = 0; i < N; ++i) {  int result = \_container.at(i).input();  if (result == -1) {  --i;  }  }  \_N = N;  }  bool increase(const unsigned long int &URN) {  for (int index = 0; index < \_N; ++index) {  if (\_container[index].\_URN == URN) {  \_container[index].\_count += 100;  break;  }  }  }  void print() {  for (auto it = \_container.begin(); it != \_container.end(); ++it) {  std::cout << "============================================\n";  it->output();  std::cout << "============================================\n";  }  }  };  int main() {  int num;  std::cin >> num;  Library lib(num);  system("cls");  lib.print();  lib.increase(1);  std::cout << "After:\n";  lib.print();  return 0;  } |

Оценка характеристик программы

В программе представлено 2 класса: Book и Library.

Book:

* Book();
* Book( … );
* input();
* output();

Library:

* Library();
* Library(int N);
* increase();
* print();

Итого: по 4 метода в каждом классе,

Отсюда видно, что основным классом является класс Book, а Library- это лишь «обёртка» для него с некоторыми функциями.

Теперь определим значение метрики NM для каждого класса:

По числу методов нельзя сказать какой класс сложнее.

Определим связность между классами объектов CBO, которая численно определяется по количеству классов, с которыми связан анализируемый класс.

Из анализа значений CBO, класс Library менее чувствителен к изменениям, чем класс Book, то есть его проще изменить, добавив или удалив что-то.

Теперь определим количество откликов на класс RFC:

не сильно различаются, а потому их тестирование не особо будут различаться по сложности.

Теперь посчитаем значение метрики LCOM (отсутствия сцепления в методах класса).

Для Book:

* Book() не делает абсолютно ничего;
* Book( … ) обращается ко всем полям класса(5 штук);
* input() обращается ко всем полям класса(5 штук);
* output() обращается ко всем полям класса(5 штук);

Таким образом для класса Book нет ни одной пары методов, которые не обращались бы к общех полей класса, и насчитывается 4 метода, которые обращаются с общим полям (всем):

т.к. при , мы приравниваем его к 0.

Для Library:

* Library() не делает ничего;
* Library(int N) обращается ко всем полям Library;
* increase() обращается ко всем полям Library;
* print() обращается ко всем полям Library;

Получаем такую же ситуацию, как и с классом Book, что говорит о том, что все классы (как же их много…) в программе тесно связаны друг с другом

Подводя итог можно сказать, что классы довольно просты, неплохо масштабируемы, слабо чувствительны к изменениям(сломаются, только если постараться), несложно тестируются и тесно связаны друг с другом