book.h

```
#ifndef BOOK_H
#define BOOK_H
#include "Attributes.h"
class Book
{
private:
    Attributes* attributes;
public:
    Book(Attributes* attrs) : attributes(attrs) {}
    Attributes* get_attributes() const {return attributes;}
};
#endif // BOOK_H
```


attributes.h

```
#ifndef ATTRIBUTES_H
#define ATTRIBUTES_H
#include <iostream>
enum class Genre
{
    ADVENTURE, CLASSICS, DETECTIVE, FANTASY, HISTORIC,
    HORROR, ROMANCE, SCIFI, UNSPECIFIED
};
inline std::ostream& operator <<(std::ostream& ostr, const Genre& genre)
{
    switch (genre)
    {
        case Genre::ADVENTURE: ostr << "adventure"; break;
        case Genre::CLASSICS:  ostr << "classics";  break;
```

```
        case Genre::DETECTIVE: ostr << "detective"; break;
        case Genre::FANTASY:   ostr << "fantasy";   break;
        case Genre::HISTORIC:  ostr << "historic";  break;
        case Genre::HORROR:    ostr << "horror";    break;
        case Genre::ROMANCE:   ostr << "romance";   break;
        case Genre::SCIFI:     ostr << "scifi";     break;

        default: ostr << "unspecified"; break;
    }
    return ostr;
}
class Attributes
{
private:
    std::string title;
    std::string last;
    std::string first;
    int    year;
    Genre  genre;
    static bool equal_ignore_case(const std::string& string1, const std::string& string2);
public:
    Attributes(const std::string& ttl, const std::string& lst, const std::string& fst,
               int yr, Genre gen)
        : title(ttl), last(lst), first(fst), year(yr), genre(gen) {}
    std::string get_title() const { return title; }
    std::string get_last()  const { return last; }
    std::string get_first() const { return first; }
    int         get_year()  const { return year; }
    Genre       get_genre() const { return genre; }
    bool is_match(const Attributes& target_attrs) const;
};
```

```
inline std::ostream& operator <<(std::ostream& ostr, const Attributes& attrs)
{
    ostr << "{TITLE: '"   << attrs.get_title()
         << "', LAST: '"  << attrs.get_last()
         << "', FIRST: '" << attrs.get_first()
         << "', YEAR: "   << attrs.get_year()
         << ", GENRE: "   << attrs.get_genre() << "}";
    return ostr;
}
#endif // ATTRIBUTES_H
```

attributes.cpp

```
#include "Attributes.h"
bool Attributes::is_match(const Attributes& target_attrs) const
{
    return equal_ignore_case(target_attrs.get_title(), title)
           && equal_ignore_case(target_attrs.get_last(),  last)
           && equal_ignore_case(target_attrs.get_first(), first)
           && (    (target_attrs.get_year()  == 0)
                   ||  (target_attrs.get_year()  == year))
           && (    (target_attrs.get_genre() == Genre::UNSPECIFIED)
                   ||  (target_attrs.get_genre() == genre));
}
bool Attributes::equal_ignore_case(const std::string& string1, const std::string& string2)
{
    if(string1.size() > string2.size()) return false;
    for (int i = 0; i < string1.size(); i++)
    {
        if (tolower(string1[i]) != tolower(string2[i]))
```

```
        {
            return false;
        }
    }
    return true;
}
```

catalogue.h

```
#ifndef CATALOGUE_H
#define CATALOGUE_H
#include <string>
#include <vector>
#include "Book.h"
#include "Attributes.h"
class Catalogue
{
private:
    std::vector<Book *> booklist;
public:
    void add(Attributes* attrs);
    std::vector<Book *> find(const Attributes& target_attrs) const;
};
#endif // CATALOGUE_H
```

catalogue.cpp

```
#include <string>
#include "Book.h"
#include "Catalogue.h"
#include "Attributes.h"
using namespace std;
void Catalogue::add(Attributes* attrs)
{
    booklist.push_back(new Book(attrs));
}
vector<Book *>Catalogue::find(const Attributes& target_attrs) const
{
    vector<Book *> matches;

    for (Book *book : booklist)
    {
        Attributes *book_attrs = book->get_attributes();
        if (book_attrs->is_match(target_attrs))
        {
            matches.push_back(book);
        }
    }
    return matches;
}
```

main.cpp

```
#include <iostream>
#include <vector>
#include <string>
#include "Book.h"
#include "Catalogue.h"
#include "Attributes.h"
using namespace std;
void fill(Catalogue& catalogue)
{
    catalogue.add(new Attributes("Life of Pi", "Martel", "Yann", 2003, Genre::ADVENTURE));
    catalogue.add(new Attributes("The Call of the Wild", "London", "Jack", 1903, Genre::ADVENTURE));
    catalogue.add(new Attributes("To Kill a Mockingbird", "Lee", "Harper", 1960, Genre::CLASSICS));
    catalogue.add(new Attributes("Little Women", "Alcott", "Louisa", 1868, Genre::CLASSICS));
    catalogue.add(new Attributes("The Adventures of Sherlock Holmes", "Doyle", "Conan", 1892, ↩
    ↪ Genre::DETECTIVE));
    catalogue.add(new Attributes("And Then There Were None", "Christie", "Agatha", 1939, ↩
    ↪ Genre::DETECTIVE));
    catalogue.add(new Attributes("Carrie", "King", "Stephen", 1974, Genre::HORROR));
    catalogue.add(new Attributes("It: A Novel", "King", "Stephen", 1986, Genre::HORROR));
    catalogue.add(new Attributes("Frankenstein", "Shelley", "Mary", 1818, Genre::HORROR));
    catalogue.add(new Attributes("2001: A Space Odyssey", "Clarke", "Arthur", 1968, Genre::SCIFI));
    catalogue.add(new Attributes("Ender's Game", "Card", "Orson", 1985, Genre::SCIFI));
}
void search(const Catalogue& catalogue, const Attributes &target_attrs)
{
    cout << endl << "Find " << target_attrs << endl;
    vector<Book *> matches = catalogue.find(target_attrs);
    if (matches.size() == 0) cout << "No matches." << endl;
    else
```

```
    {
        cout << "Matches:" << endl;
        for (Book *book : matches)
        {
            cout << "  " << *book->get_attributes() << endl;
        }
    }
}
void test(const Catalogue& catalogue)
{
    Attributes target_attrs1("Life of Pi", "Martel", "Yann", 2003, Genre::ADVENTURE);
    search(catalogue, target_attrs1);
    Attributes target_attrs2("", "King", "", 0, Genre::HORROR);
    search(catalogue, target_attrs2);
    Attributes target_attrs3("1984", "Orwell", "George", 0, Genre::CLASSICS);
    search(catalogue, target_attrs3);
    Attributes target_attrs4("", "", "", 1960, Genre::ROMANCE);
    search(catalogue, target_attrs4);
    Attributes target_attrs5("", "", "", 1960, Genre::UNSPECIFIED);
    search(catalogue, target_attrs5);
    Attributes target_attrs6("", "", "", 0, Genre::SCIFI);
    search(catalogue, target_attrs6);
    Attributes target_attrs7("", "", "", 0, Genre::UNSPECIFIED);
    search(catalogue, target_attrs7);
}
int main()
{
    Catalogue catalogue;
    fill(catalogue);
    test(catalogue);
}
```

Uruchomienie

```
Find {TITLE: 'Life of Pi', LAST: 'Martel', FIRST: 'Yann', YEAR: 2003, GENRE: adventure}
Matches:
  {TITLE: 'Life of Pi', LAST: 'Martel', FIRST: 'Yann', YEAR: 2003, GENRE: adventure}

Find {TITLE: '', LAST: 'King', FIRST: '', YEAR: 0, GENRE: horror}
Matches:
  {TITLE: 'Carrie', LAST: 'King', FIRST: 'Stephen', YEAR: 1974, GENRE: horror}
  {TITLE: 'It: A Novel', LAST: 'King', FIRST: 'Stephen', YEAR: 1986, GENRE: horror}

Find {TITLE: '1984', LAST: 'Orwell', FIRST: 'George', YEAR: 0, GENRE: classics}
No matches.

Find {TITLE: '', LAST: '', FIRST: '', YEAR: 1960, GENRE: romance}
No matches.

Find {TITLE: '', LAST: '', FIRST: '', YEAR: 1960, GENRE: unspecified}
Matches:
  {TITLE: 'To Kill a Mockingbird', LAST: 'Lee', FIRST: 'Harper', YEAR: 1960, GENRE: classics}

Find {TITLE: '', LAST: '', FIRST: '', YEAR: 0, GENRE: scifi}
Matches:
  {TITLE: '2001: A Space Odyssey', LAST: 'Clarke', FIRST: 'Arthur', YEAR: 1968, GENRE: scifi}
  {TITLE: 'Ender's Game', LAST: 'Card', FIRST: 'Orson', YEAR: 1985, GENRE: scifi}

Find {TITLE: '', LAST: '', FIRST: '', YEAR: 0, GENRE: unspecified}
Matches:
  {TITLE: 'Life of Pi', LAST: 'Martel', FIRST: 'Yann', YEAR: 2003, GENRE: adventure}
  {TITLE: 'The Call of the Wild', LAST: 'London', FIRST: 'Jack', YEAR: 1903, GENRE: adventure}
  {TITLE: 'To Kill a Mockingbird', LAST: 'Lee', FIRST: 'Harper', YEAR: 1960, GENRE: classics}
```

```
  {TITLE: 'Little Women', LAST: 'Alcott', FIRST: 'Louisa', YEAR: 1868, GENRE: classics}
  {TITLE: 'The Adventures of Sherlock Holmes', LAST: 'Doyle', FIRST: 'Conan', YEAR: 1892, GENRE: detective}
  {TITLE: 'And Then There Were None', LAST: 'Christie', FIRST: 'Agatha', YEAR: 1939, GENRE: detective}
  {TITLE: 'Carrie', LAST: 'King', FIRST: 'Stephen', YEAR: 1974, GENRE: horror}
  {TITLE: 'It: A Novel', LAST: 'King', FIRST: 'Stephen', YEAR: 1986, GENRE: horror}
  {TITLE: 'Frankenstein', LAST: 'Shelley', FIRST: 'Mary', YEAR: 1818, GENRE: horror}
  {TITLE: '2001: A Space Odyssey', LAST: 'Clarke', FIRST: 'Arthur', YEAR: 1968, GENRE: scifi}
  {TITLE: 'Ender's Game', LAST: 'Card', FIRST: 'Orson', YEAR: 1985, GENRE: scifi}

Process returned 0 (0x0)   execution time : 0.029 s
Press any key to continue.
```