BLG252E – OBJECT ORIENTED PROGRAMMING

The First Practice Session

11.03.2013

Question 1

Taken from Midterm #1 (2012 Spring)

Book Class

- You need to design a class to model books (**Book**).
  - Each book is represented by an integer **isbn**, and two strings for the **title** of the book and the **author**, which are given during the creation of a book.
  - If the **isbn** information is not given, its value is set to **0**.
  - The information of a book can be printed on the screen.

Bookcase Class

- You need to design a class for a **Bookcase** object which will contain a given number (**maxCap**) of **books.**
  - The class will hold a **dynamic array of its books** and a **pointer (b_ptr)** to this array.
  - The current capacity (**curCap**) information will be stored as the number of books that currently exist in the bookcase.
  - It will be possible to check whether a bookcase has reached its capacity limit (**maxCap**) by checking a flag (**full)** maintained in the class.
  - A bookcase can have more than one copies of a book.
  - **printAll, addBook** and **findBook** services are given for a bookcase.

To do

- Please design only the **required** classes with all **attributes** being **private** and implement them in C++.
- Provide **only** the required and necessary methods.
- Make sure that your classes are **efficiently** and **properly designed for public use.**

Sample test program

```
int main(){
    //Some books are created
    //Title: "Artificial Intelligence, A Modern Approach", Author: "Stuart Russell", ISBN: 131038052
    Book b1(string("Artificial Intelligence, A Modern Approach"),string("Stuart Russell"),131038052);
    //Title: "Modern Operating Systems", Author: "Andrew S. Tanenbaum", ISBN: 136006639
    Book b2(string("Modern Operating Systems"),string("Andrew S. Tanenbaum"), 136006639);
    //Title: "Introduction to Algorithms", Author: "Thomas Cormen", ISBN is not given: it will be 0
    Book b3(string("Introduction to Algorithms"),string("Thomas Cormen"));

    //The array elements are assigned to these books
    Book *bookArray = new Book[3];
    bookArray[0] = b1;
    bookArray[1] = b2;
    bookArray[2] = b3;

    // A Bookcase is created to include 20 books, the given array of books is used for the first 3 books
    // Max capacity: 20, a book array pointer is given to create books in the bookcase
    // the array size: 3
    Bookcase bc1(20,bookArray,3);

    //Prints the information about all the books in the bookcase
    bc1.printAll();
```

Sample test program

```
    //Another copy of "Introduction to Algorithms" book is created
    Book b4(b3);
    b4.print();

    delete [] bookArray;

    // A second copy of "Introduction to Algorithms" is added into the bookcase
    if(bc1.addBook(b4))
         cout << "The given book has been added.." << endl;
    else
         cout << "The capacity limit is reached.." << endl;

    // Another bookcase is created
    Bookcase bc2(bc1);
    bc2.printAll();

    // Prints the information of the book with ISBN: 131038052, or an error message if it does not exist
    bc1.findBook(131038052);

    return 0;
}
```

Desired Output

```
---------------------------------------
Printing Bookcase..
Book information
        Book ISBN: 131038052
        title: Artificial Intelligence, A Modern Approach
        Author: Stuart Russell
Book information
        Book ISBN: 136006639
        title: Modern Operating Systems
        Author: Andrew S. Tanenbaum
Book information
        Book ISBN: 0
        title: Introduction to Algorithms
        Author: Thomas Cormen
---------------------------------------
Book information
        Book ISBN: 0
        title: Introduction to Algorithms
        Author: Thomas Cormen
The given book has been added..
---------------------------------------
Printing Bookcase..
Book information
        Book ISBN: 131038052
        title: Artificial Intelligence, A Modern Approach
        Author: Stuart Russell
Book information
        Book ISBN: 136006639
        title: Modern Operating Systems
        Author: Andrew S. Tanenbaum
Book information
        Book ISBN: 0
        title: Introduction to Algorithms
        Author: Thomas Cormen
Book information
        Book ISBN: 0
        title: Introduction to Algorithms
        Author: Thomas Cormen
---------------------------------------
Book information
        Book ISBN: 131038052
        title: Artificial Intelligence, A Modern Approach
        Author: Stuart Russell
```

Solution

“Book” Class

```
class Book{
   int isbn;
   string title, author;
public:
   Book(){}
   // constructor
   Book(const string&, const string&, int=0);
   // getters and setters
   int getISBN() const;
   void setISBN(int);
   string getTitle() const;
   void setTitle(const string&);
   string getAuthor() const;
   void setAuthor(const string&);
   // print method
   void print() const;
};
```

“Book” Class

```
// constructor with parameters
Book::Book(const string& n_title, const string& n_author, int n_isbn){
   isbn = n_isbn;
   title = n_title;
   author = n_author;
}

// print method body
void Book::print() const {
   cout << "Book information" << endl
   << "\t Book ISBN: " << isbn << endl
   << "\t title: " << title << endl
   << "\t Author: " << author << endl;
}
```

“Book” Class

```
// Make sure that your classes are efficiently and properly designed
for public use: getter and setter methods are necessary
int Book::getISBN() const {return isbn;}

void Book::setISBN(int isbn_in) {isbn = isbn_in;}

string Book::getTitle() const {return title;}

void Book::setTitle(const string& title_in) {title = title_in;}

string Book::getAuthor() const {return author;}

void Book::setAuthor(const string& author_in) {author = author_in;}
```

“Bookcase” Class

```
class Bookcase{
    Book *b_ptr;
    const int maxCap;
    int curCap;
    bool full;
public:
    Bookcase(int=100);
    Bookcase(int, const Book*, int);
    Bookcase(const Bookcase&); // copy constructor is needed
    // operator= is needed but not included for this question
    bool addBook(Book&);
    void findBook(int) const;
    void printAll() const;
    // destructor is needed
    ~Bookcase();
};
```

“Bookcase” Constructors and Destructor

```
Bookcase::Bookcase(int m_capacity): maxCap(m_capacity){
    curCap = 0;
    full = false;
    b_ptr = new Book[m_capacity];
}

Bookcase::Bookcase(int m_capacity,const Book* in_ptr,int in_arraySize): maxCap(m_capacity){
    curCap = in_arraySize;
    full = false;
    b_ptr = new Book[m_capacity];
    for(int i=0;i<curCap;i++)
          b_ptr[i]=in_ptr[i];
}

Bookcase::~Bookcase(){
    delete[] b_ptr;
}
```

“Bookcase” Copy Constructor

```
Bookcase::Bookcase(const Bookcase& in_b):maxCap(in_b.maxCap){
     curCap = in_b.curCap;
     full = in_b.full;
     b_ptr = new Book[in_b.maxCap];
     for(int i=0;i<maxCap;i++)
          b_ptr[i]=in_b.b_ptr[i];
}
```

“Bookcase” other Methods

```
bool Bookcase::addBook(Book& in_b){
     if(full) return false;
     b_ptr[curCap] = in_b;
     curCap++;
     if(curCap == maxCap) full = true;
     return true;
}
void Bookcase::findBook(int in_isbn) const{
     for(int i = 0; i < curCap; i++){
          if(b_ptr[i].getISBN() == in_isbn){
              b_ptr[i].print();
              return;
          }
     }
     cout << "Book has not been found in the Bookcase" << endl;
}
void Bookcase::printAll() const{
     cout << "----------------------------------------" << endl;
     cout << "Printing Bookcase.." << endl;
     for(int i = 0; i < curCap; i++)
          b_ptr[i].print();
     cout << "----------------------------------------" << endl;
}
```

Question 2

Taken from Midterm #1 (2012 Spring)

Question

- Analyze run-time behavior of the following code and write **the output** generated by each line of the code. If there is **no output** for a given line, **leave** the related cell **empty**.

```
class Aclass{
    int i;
public:
    Aclass(): i(0) {
        cout << "Aclass(): i= " << i << endl;
    }
    Aclass(int new_i): i(new_i) {
        cout << "Aclass(int): i= " << i << endl;
    }
    ~Aclass() {
        cout  << "~Aclass()" << "i= " << i << endl;
    }
    Aclass(const Aclass &in_obj) {
        i = in_obj.i;
        cout << "Aclass(const &)"<< "i= " << i << endl;
    }
};
```

```
int main(){
    Line 1: Aclass obj1;
    Line 2: Aclass *obj2_ptr;
    Line 3: obj2_ptr = new Aclass(5);
    Line 4: Bclass obj3;
    Line 5: obj3.func1(*obj2_ptr);
    Line 6: obj3.func2();
    Line 7: Bclass obj4 = obj3;
    Line 8: delete obj2_ptr;
    Line 9: return 0;
}
```

```
class Bclass{
    Aclass a;
public:
    Bclass() {
        cout << "Bclass() default" << endl;
    }
    ~Bclass() {
        cout << "~Bclass()" << endl;
    }
    void func1(Aclass obj) {
        cout << "func1" << endl;
    }
    void func2() {
        cout << "func2" << endl;
        Aclass obj(9);
    }
    Bclass(const Bclass &in_b):a(in_b.a) {
        cout << "Bclass(const &)" << endl;
    }
};
```

Solution

Line 1:	Aclass(): i= 0
Line 2:	-
Line 3:	Aclass(int): i= 5
Line 4:	Aclass(): i= 0 Bclass() default
Line 5:	Aclass(const &)i= 5 func1 ~Aclass()i= 5
Line 6:	func2 Aclass(int): i= 9 ~Aclass()i= 9
Line 7:	Aclass(const &)i= 0 Bclass(const &)
Line 8:	~Aclass()i= 5
Line 9:	~Bclass() ~Aclass()i= 0 ~Bclass() ~Aclass()i= 0 ~Aclass()i= 0